НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

«До захисту допущено»
В. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
з напряму підготовки 6.040303 «Системний аналіз»
на тему: «Методи передбачення часових рядів на прикладі вартості акцій»

Виконав:
студент IV курсу, групи КА-51
Олексієнко Ганна Олегівна _____

Керівник:
асистент Кухарєв С.О. _____

Консультант з економічного розділу:
доцент, к.е.н. Шевчук О.А. _____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А. Є. _____

Рецензент:
доцент, к.т.н. Безносик О.Ю. _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.040303 «Системний аналіз» («Системний аналіз і управління»)

ЗАТВЕРДЖУЮ

В.о.завідувача кафедри

_____ О.Л. Тимощук

«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Олексієнко Ганні Олегівні

1. Тема роботи «**Методи передбачення часових рядів на прикладі вартості акцій**», керівник роботи Кухарєв Сергій Олександрович, асистент, затверджені наказом по університету від «25»травня 2019 р. №1353с.

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи _____

_____

4. Зміст роботи _____

_____

_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А., доцент		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка

Студент _____ (підпис) _____ (ініціали, прізвище)

Керівник роботи _____ (підпис) _____ (ініціали, прізвище)

РЕФЕРАТ

Дипломна робота: 154 с., 19 рис., 12 табл., 3 додатки, 13 джерел.

ЧАСОВИЙ РЯД, АВТОРЕГРЕСІЯ, ЗГЛАДЖУВАННЯ, НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ

Об’єкт дослідження – дані про акції компанії Google з офіційного сайту фондової біржі NASDAQ за 5 років, починаючи з 1 квітня 2014 року.

Предмет дослідження – методи прогнозування часових рядів: традиційні моделі авторегресії ARMA, ARIMA, експоненційне згладжування, а також методи інтелектуального аналізу даних з використанням штучних нейронних мереж та глибинного навчання – згорткові та рекурентні нейронні мережі.

Мета роботи – проаналізувати предмет дослідження, виявити параметри впливу на ефективність та точність деяких моделей, що використовуються для аналізу та прогнозування вартості акцій.

Метод дослідження – розгляд та аналіз методів передбачення за обраними метриками.

Актуальність – надання можливість точного передбачення вартості акцій, що сприятиме ймовірному отриманню фінансового прибутку компаніями, урядом або іншими гравцями на фондових біржах.

Було проведено порівняльний аналіз розглянутих методів прогнозування. Шляхи подальшого розвитку предмету дослідження – методи ансамблевого навчання нейронних мереж, створення нових ознак для нейронних мереж, збір більшого датасету для прогнозування.

ABSTRACT

Thesis: 154 p., 19 fig., 12 tabl., 3 appendixes, 13 sources

TIME SERIES, AUTOREGRESSION, SMOOTHING, NEURAL NETWORKS, CONVOLUTIONAL NEURAL NETWORKS, RECURRENT NEURAL NETWORKS

The object of this research is time series of Google LLC share prices from the official NASDAQ Stock Exchange data starting April 1, 2014.

The subject of the research is the methods of prediction of time series: traditional models of auto-regression, smoothing techniques and machine learning methods with the use of artificial neural networks and deep learning.

The purpose of the work is to analyze the subject of the research, to find out the parameters which influence the efficiency and accuracy of models for analysis and forecast the share prices.

The methods of research include the consideration and analysis of prediction methods using specific metrics.

The relevance of the topic assumes that the accurate prediction at financial market may contribute to the financial benefits for companies, government and other players on stock exchanges.

A comparative analysis of the considered forecasting methods was conducted. The further development of the subject of research includes ensemble learning methods for neural networks, feature engineering, collecting a larger data set for forecasting.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ **9**

ВСТУП **10**

1.1. Аналіз існуючих підходів до вирішення задачі 12

1.1.2 Технічний аналіз 13

1.1.3 Технологічний аналіз 13

1.2. Поняття часового ряду 14

1.3. Дані для дослідження 17

1.4. Опис метрик 18

1.4.1 Середня абсолютна помилка (MAE) 19

1.4.2 Середньоквадратична помилка (MSE) 19

Висновки до розділу 1 20

РОЗДІЛ 2 КЛАСИЧНІ МАТЕМАТИЧНІ МЕТОДИ ПРОГНОЗУВАННЯ ВАРТОСТІ АКЦІЙ **21**

2.1 Вступ до розділу 2 21

2.2 Методи згладжування 21

2.2.1 Просте ковзне середнє 21

2.2.2 Зважене ковзне середнє 22

2.2.3 Експоненціальне ковзне середнє 23

2.2.4 Подвійне експоненціальне ковзне середнє 24

2.2.5 Потрійне експоненціальне ковзне середнє 25

2.3 Методи авторегресії 26

2.3.1 Проста модель авторегресії (AR - Autoregression) 27

2.3.2 Модель авторегресії — ковзного середнього (ARMA - Autoregressive moving average) 28

2.3.3 Модель авторегресії — інтегрованого ковзного середнього (ARIMA - Autoregressive integrated moving average) 28

2.3.4 Знаходження значень параметрів 30

Висновки до розділу 2 31

РОЗДІЛ 3 МАТЕМАТИЧНІ МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ ПРОГНОЗУВАННЯ ВАРТОСТІ АКЦІЙ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ 32

3.1 Вступ до розділу 3 32

3.2 Загальні відомості про нейронні мережі 32

3.3. Вхідні дані 37

3.4 Багатошаровий перцептрон Румельхарта (Multilayer Perceptron - MLP) 39

3.4 Згорткова нейронна мережа (Convolutional Neural Network - CNN) 40

3.5 Рекурентна нейронна мережа (Recurrent Neural Network - RNN) 41

Висновки до розділу 3 42

РОЗДІЛ 4 РЕЗУЛЬТАТИ РЕАЛІЗАЦІЇ МЕТОДІВ ПРОГНОЗУВАННЯ НА ДАНИХ 43

4.1 Вступ до розділу 4 43

4.2 Реалізація методів згладжування 43

4.3 Реалізація методів авторегресії 48

4.4 Реалізація MLP 49

4.5 Реалізація згорткових нейронних мереж (CNN) 52

4.5 Реалізація рекурентних нейронних мереж (RNN) 54

Висновки до розділу 4 57

РОЗДІЛ 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ 58

5.1 Постановка завдання проектування 58

5.2. Обґрунтування функцій програмного продукту 59

5.3 Обґрунтування системи параметрів програмного продукту 62

5.4. Аналіз експертного оцінювання параметрів 66

5.5. Аналіз рівня якості варіантів реалізації функцій 71

25.6. Економічний аналіз варіантів розробки програмного продукту 72

5.7. Вибір кращого варіанту ПП техніко-економічного рівня 77

ВИСНОВКИ 79

ПЕРЕЛІК ПОСИЛАНЬ 80

ДОДАТОК А ВХІДНІ ДАНІ ВАРТОСТІ АКЦІЙ 82

ДОДАТОК Б ТЕКСТ ПРОГРАМИ 96

ДОДАТОК В СТАТТІ 125

ДОДАТОК Г ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ 137

ПЕРЕЛІК СКОРОЧЕНЬ

MAE – Mean Absolute Error

MSE – Mean Squared Error

MA - Moving Average

EMA – Exponential Moving Average

ANN– Artificial Neural Networks

CNN – Convolutional Neural Networks

RNN – Recurrent Neural Networks

LSTM – Long Short-Term Memory

ARMA – Autoregressive moving average

ARIMA – Autoregressive integrated moving average

AIC – Інформаційний критерій Акайке

ВСТУП

Фондовий ринок є однією з найважливіших сфер ринкової економіки, оскільки він надає компаніям доступ до капіталу, дозволяючи інвесторам купувати акції в компанії. Купуючи акції, інвестори можуть отримати гроші для майбутнього розвитку компаній. Проте не всі інвестори успішно отримують прибуток від своїх інвестицій. Це відбувається тому, що ціна акцій постійно коливається, і в будь-який момент ціна акції може впасти нижче ціни, за якою вона була куплена. Тому передбачення того, як буде поводитись фінансовий ринок, є одним з найважчих завдань в економіці. У передбаченні варто врахувати багато факторів - фізичні, психологічні, раціональні та ірраціональні поведінки тощо. Всі ці аспекти приводять до висновку для того, що ціни на акції є дуже нестійкими, і їх дуже важко передбачити з високим ступенем точності. Проте ця задача є актуальною для всього світу та для всієї міжнародної економіки, оскільки можливість точного передбачення вартості акцій тісно пов’язано з отриманням фінансового прибутку компаній, уряду або особистого капіталу, та формуванням більш раціональної фінансової поведінки. Точне прогнозування вартості активів на біржі також дозволить зменшити інвестиційний ризик та захистити інвестиційні прибутки від волатильності ринку.

Об'єктом дослідження є статистичні дані про акції компанії Google з офіційного сайту фондової біржі NASDAQ за 5 років, починаючи з 1 квітня 2014 року.

Предметом дослідження є методи прогнозування часових рядів.

Мета роботи – проаналізувати предмет дослідження, дослідити відомі та найбільш поширені методи для прогнозування вартості акцій на фінансових ринках, виявити параметри впливу на ефективність та точність деяких моделей,

що використовуються для аналізу та прогнозування вартості акцій, щоб зробити висновки з проведеного дослідження.

Методом дослідження є побудова прогнозних моделей та аналіз отриманих результатів.

Пояснювальна записка складається з п’яти розділів. У першому розділі досліджуються існуючі підходи до передбачення вартості акцій. У другому та третьому розділі розглянуто ряд математичних моделей, за допомогою яких будують прогнози. Четвертий розділ описує архітектуру розробленої програми, та остаточні порівняльні результати роботи всіх моделей. П’ятий розділ являє собою економічну частину, в якій розробляється кошторис витрат на розробку.

РОЗДІЛ 1 ПІДХОДИ ДО ЗАДАЧІ ПРОГНОЗУВАННЯ ВАРТОСТІ АКЦІЙ

1.1. Аналіз існуючих підходів до вирішення задачі

В історичній перспективі багато людей намагалися заробити прибуток на фінансових ринках через прогнозування майбутньої ціни товарів, запасів, курсів валют, опціонів. Усі розроблені методики можна поділити на три категорії.

1.1.1 Фундаметальний аналіз

Фундаментальний аналіз передбачає поглиблений аналіз результатів діяльності та прибутковості компанії для визначення ціни акцій. Він використовує базову ринкову інформацію для прогнозування майбутнього переміщення активу. Інвестори розглядають таку інформацію, як дохід, пропозиція, попит, операційна маржа, загальні економічні умови, конкуренція інших компаній. Фундаментальний аналіз пов'язаний з макроекономічними та політичними факторами, які можуть вплинути на майбутньої ціни фінансового активу. Перевагою такого методу є його систематичний підхід та його здатність передбачати зміни перед тим, як вони з'являться на графіках. Інвестори вивчають компанії та порівнюють їх з поточною економічною ситуацією. Проте, все важче формалізувати всі ці знання для автоматизації, і інтерпретація цих знань може бути суб'єктивною[1]. В даній роботі фундаментальний аналіз не розглядається.

1.1.2 Технічний аналіз

Технічним аналізом є вивчення історичних цін і закономірностей з метою прогнозування майбутні ціни. Він широко використовується у світі фінансів. Використовуючи статистику цін та обсягів продажу, технічний аналітик вивчає графіки для виявлення тенденцій та прогнозування майбутніх рухів активів. Вважається, що тенденції базуються на питаннях попиту та пропозиції, які часто мають циклічні або помітні закономірності. Існують технічні індикатори, наприклад, ковзне середнє, які можна формалізувати або використовувати як вхідні дані для нейронних мереж. У деяких простих системах торгівлі, перетин кривої вартості акції та ковзного середнього або двох ковзних середніх використовуються як сигнали до купівлі або продажу акції. Технічний аналіз спирається на припущення, що історія повторюється і що майбутній ринковий напрямок може бути визначений шляхом вивчення минулих цін. Незважаючи на широке використання, технічний аналіз піддається критиці, оскільки він дуже суб'єктивний, бо різні люди можуть інтерпретувати графіки різними способами.

1.1.3 Технологічний аналіз

З диджиталізацією інформації прогноз на фінансовому ринку перейшов у технологічну сферу. Розроблені методи прогнозування часових рядів були застосовані для проблеми передбачення коливань цін на фінансовому ринку.

Класичний аналіз часових рядів, наприклад, експоненційне згладжування, процедура Голта-Вінтерса, модель авторегресії - ковзного середнього (ARMA), інтегрована модель авторегресії - ковзного середнього (ARIMA) широко використовуються на сучасний ринках.

Проте, в останні розробляються методи інтелектуального аналізу даних з використанням штучних нейронних мереж та глибинного навчання, а також

гібридний підхід поєднання класичних методів з методами інтелектуального аналізу даних. Враховуючи здатність нейронних мереж вивчати нелінійні, хаотичні системи, можливо, можна перевершити традиційний аналіз та інші комп'ютерні методи.

Із розвитком фінансових ринків, з’являється все більше способів для передбачення вартості акцій. Різноманітні методи були запропоновані та використані з різними результатами. Проте, жоден метод або комбінація методів не були достатньо успішними, щоб стати провідним способом. Ця робота дає змогу проаналізувати класичні та сучасні методів прогнозування цін на акцій з акцентом на те, чому вони є недостатніми, та як нейронні мережі використовуються для їх вдосконалення.

1.2. Поняття часового ряду

В цій роботі досліджується часовий ряд. Нехай $y_1, y_2, ...y_T$ - значення спостережень за економічним процесом протягом Т періодів. Ця послідовність є числовими значеннями, кожне з яких має відповідний індекс, який залежить від номера періоду, в який він спостерігався. Така послідовність, записана у порядку зростання індексу, називається часовим рядом $\{Y_T\}$. Отже, часовим рядом є послідовність даних дискретного часу.

Специфіка часових рядів полягає в тому, що дуже важливим, якщо не критичним вважається спосіб впорядкування даних в часі, оскільки зазвичай спостереження не є незалежними, тоді як більшість інших статистичних теорій стосується випадкових вибірок незалежних спостережень.

Часовий ряд можна представити за допомогою чотирьох компонентів: тренд, сезонні коливання, циклічні зміни та нерегулярні фактори.

Класичний підхід до побудови моделі часового ряду полягає у розкладі його на декілька компонент, кожна з яких аналізується специфічними для неї методами:

$$y_t = tr_t + s_t + c_t + r_t \ . \tag{1.1}$$

Тренд tr_t — загальна тенденція при різнонаправленому русі, визначена загальною спрямованістю змін показників часового ряду. В широкому розумінні, тренд — це тривала зміна рівня середнього випадкового процесу. Тренд може показувати зростання або зниження значень часових рядів, наприклад, ціни, дані експорту та імпорту відображають зростаючі тенденції з часом. Загалом, аналіз часового ряду починається з виділення тренду.

Серед чинників,що визначають регулярні коливання ряду, розрізняють сезонії та циклічні.

Сезонні коливання s_t, мають періодичний або близький до нього характер упродовж одного року. Це короткочасні рухи, що відбуваються в даних через сезонні фактори. Наприклад, ціни на сільгосппродукцію влітку вищі, ніж взимку, рівень безробіття в курортних містах у зимовий період зростає відносно до літнього.

Циклічні (кон'юнктурні) коливання c_t схожі на сезонні, але виявляються на триваліших інтервалах часу. Циклічні коливання пояснюються дією довготермінових циклів економічної, демографічної або астрофізичної природи: динаміка показника містиме характерні зміни, що повторюються з однаковою циклічністю.

Тренд, сезонна і циклічна компоненти не є випадковими, тому їх називають систематичними або детермінованими компонентами часового ряду.

Одним з параметрів, від яких залежить детермінований компонент часового ряду, є час.

Нерегулярні коливання r_t - це раптові зміни, що відбуваються в часових рядах, які навряд чи будуть повторені. Вони є складовими часових рядів, які не можна пояснити тенденціями, сезонними або циклічними рухами. Ці варіації іноді називають залишковими або випадковими компонентами. Ці варіації, хоча й випадкові за своїм характером, можуть спричинити постійну зміну тенденцій, сезонних і циклічних коливань протягом майбутнього періоду. Повені, пожежі, землетруси, революції, епідемії, страйки тощо - корінні причини таких порушень.

В сучасній статистичній теорії існує багато різноманітних методів прогнозування економічної інформації. Модель часових рядів якраз враховує минулу поведінку даної змінної і використовує цю інформацію для прогнозування її майбутньої поведінки. Особливістю прогнозування часових рядів є те, що аналізуються лише дані спостережень без додаткової інформації, без аналізу впливу зовнішніх сил. Основний напрямок розвитку теорії часових рядів є моделювання і аналіз процесу r_t .

Представлення часового ряду за формулою (1.1) називається адитивною моделлю представлення часового ряду. Якщо замість поточних значень ряду використовувати їх логарифми, то таке представлення зветься мультиплікативною моделлю:

$$ln\, y_t = ln\, tr_t + ln\, s_t + ln\, c_t + ln\, r_t \text{ або } y_t = tr_t \cdot s_t \cdot c_t \cdot r_t \text{ [2].}$$

1.3. Дані для дослідження

Для виконання роботи було взято дані про акції компанії Google з офіційного сайту NASDAQ за 5 років, починаючи з 1 квітня 2014 року.
NASDAQ (National Association of Securities Dealers Automated Quotation — Автоматизовані котирування Національної асоціації дилерів цінних паперів) — американський біржовий ринок, що спеціалізується на акціях високотехнологічних компаній (виробництво електроніки, програмного забезпечення тощо). Одна з трьох основних фондових бірж США, на якій зареєстровано понад 3,3 тис. компаній. Учасниками Nasdaq є провідні компанії в таких напрямках бізнесу, як високі технології, торгівля, телекомунікації, фінансові послуги, транспорт, ЗМІ, біотехнології.

Датасет представляє з себе щоденну інформацію про ціну на акцій на початку та в кінці дня (рис. 1.1), максимальну та мінімальну вартість акції за день, та кількість проданих акцій (рис. 1.2).

	date	close	volume	open	high	low
0	5/10/19	1164.27	1314546	1162.38	1172.600	1142.50
1	5/9/19	1162.38	1185973	1166.27	1169.660	1150.85
2	5/8/19	1166.27	1309514	1174.10	1180.424	1165.74
3	5/7/19	1174.10	1551368	1189.39	1190.440	1161.04
4	5/6/19	1189.39	1563943	1185.40	1190.850	1166.26

Рисунок 1.1 – Вхідні дані

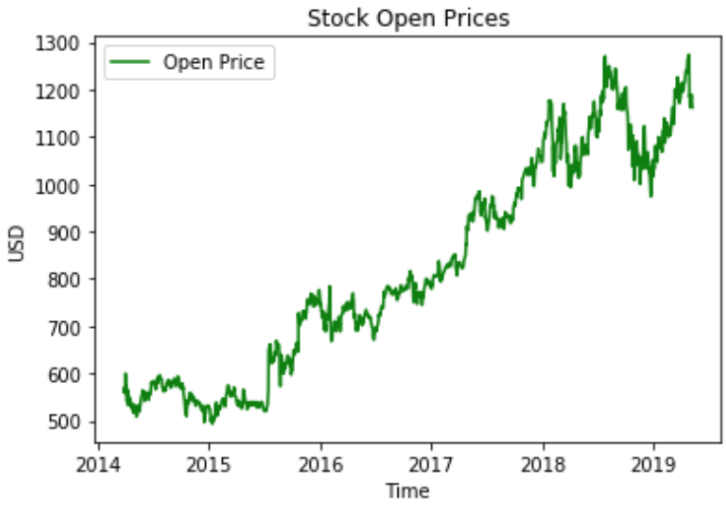


Рисунок 1.2 – Графік цін на початку дня

1.4. Опис метрик

Загалом, для аналізу проблеми передбачення цін використовуються статистичні та нестатистичні метрики.

Нестатистичні показники результативності включають річний прибуток конкретної моделі, а також коефіцієнт попадання чи кількість раз, коли модель правильно передбачила, що ринок буде йти вгору чи вниз.

Для аналізу використовують різноманітні статистичні показники, наприклад середня абсолютна похибка (MAE), середньоквадратичний квадрат (RMSE), середня квадратична помилка передбачення (MSPE), коефіцієнт кореляції та прямокутний кореляційний коефіцієнт та мінімальна остаточна помилка передбачення Акайке.

В даній роботі всі моделі та методи аналізуються за допомогою метрик MAE та MSE.

1.4.1 Середня абсолютна помилка (MAE)

MAE вимірює середню абсолютну величину помилок у наборі прогнозів для безперервних змінних.

Припустимо, що $\hat{y}_t$ - прогноз значення часового ряду у періоді t, тоді метрика задається формулою:

$$MAE = \sum_t \left| y_t - \hat{y}_t \right| .$$

MAE є лінійною оцінкою, що означає, що всі індивідуальні відмінності зважуються однаково в середньому.

1.4.2 Середньоквадратична помилка (MSE)

MSE є різницею між прогнозом і відповідними спостережуваними значеннями кожна квадратична, а потім усереднена по вибірці. Оскільки помилки підносяться до квадрату перед тим, як вони усереднюються, MSE надає відносно високу вагу великим похибкам. Це означає, що MSE є найбільш корисним, коли великі помилки особливо небажані, що відповідає цілям даної роботи.

Припустимо, що $\hat{y}_t$ - прогноз значення часового ряду у періоді t, тоді метрика задається формулою:

$$MSE = \sum_t (y_t - \hat{y}_t)^2 .$$

Обидві метрики, MSE та MAE, можуть набувати значення від 0 до ∞, і не враховують напрямок помилок. Чим менше значення приймає показник, тим точнішим є прогноз.

Висновки до розділу 1

У розділі 1 розглянуто існуючі підходи до прогнозування фінансових даних, а саме базові принципи фундаментального, технічного та технологічного аналізу. На основі цього було обрано технологічний аналіз в якості напряму поточного дослідження.

Також було розглянуто поняття часового ряду, де було наведено детальний опис кожної компоненти. В якості часового ряду у цій роботі було обрано фінансовий часовий ряд – а саме вартість акцій компанії на американській біржі NASDAQ протягом більш 5 років.

Завданням даної роботи є побудова прогнозу вартості акцій, отже є вимога мати метрики для оцінювання. Серед усіх метрик було обрано середню абсолютну помилку та середньоквадратичну помилку для оцінки якості прогнозу.

РОЗДІЛ 2 КЛАСИЧНІ МАТЕМАТИЧНІ МЕТОДИ ПРОГНОЗУВАННЯ ВАРТОСТІ АКЦІЙ

2.1 Вступ до розділу 2

В цьому розділі буде прогнозуватись середня ціна акції за допомогою методів усереднення, згладжування та авторегресійних методів.

2.2 Методи згладжування

Згладжування - це важливий і широко поширений метод прогнозування фінансових ринків. Як правило, різні методи згладжування базуються на концепції ковзних середніх. Це допомагає зменшити вплив випадкового компонента у часовому ряді. Цей розділ представить 5 методів згладжування, які зазвичай зустрічаються при прогнозуванні фінансових даних. Основне припущення цих методів полягає у тому, що коливання минулих значень являють собою випадкові відхилення від деякої плавної кривої, яка може бути екстрапольована для створення прогнозу.

2.2.1 Просте ковзне середнє

Термін ковзне середнє означає, що набір значень, що усереднюють, безупинно рухається в часі. Ковзне середнє відображає тенденцію зміни цін і згладжує їхні несуттєві коливання. Оскільки ковзне середнє є середнім значенням цін у минулому, графіки ковзних середніх “відстають” від поточних змін у часовому ряді. На ринках з чітко вираженим трендом ковзні середні

показують добрий результат, проте на ринках, де немає чітко вираженої тенденції, метод дає великі похибки. Формально метод описується так:

$$\hat{y_t} = \frac{1}{N} \sum_{i=1}^{N} y_{k-i},$$

де N - число попередніх моментів часу, що було взяти до уваги при побудові прогнозу,

y_{k-i} - реальні значення показника в момент часу t_{k-i}.

2.2.2 Зважене ковзне середнє

Метод зваженого ковзного середнього розширює ідею простого ковзного середнє, оскільки кожен компонент часового ряду зрівноважується відповідними ваговими коефіцієнтами, тобто:

$$\hat{y_t} = \frac{1}{N} \sum_{i=1}^{N} w_i y_{k-i},$$

де N - число попередніх моментів часу, що було взяти до уваги при побудові прогнозу,

y_{k-i} - реальні значення показника в момент часу t_{k-i},

w_i - ваговий коефіцієнт для i-того компоненту ряду[3].

Перевагою зважених коефіцієнтів є те, що в результаті отримана оцінка тренду є набагато гладше. Замість того, щоб кожне вхідне спостереження різко

змінювало значення середнього, значення спостережень можуть бути зважені за допомогою коефіцієнтів.

2.2.3 Експоненціальне ковзне середнє

Експоненціальне ковзне середнє (Exponential Moving Average - EMA) походить з ідеї зваженого ковзного середнього, але у порівнянні з простим ковзним середнім, більший акцент робиться на останні точки даних і результуючі усереднені значення ближче до фактичних спостережень за набором даних. Вагові коефіцієнти зменшуються експоненціально, в результаті чого акцент падає на останні спостереження, хоча не відкидаючи старі. Також припускається, що в цьому процесі не існує систематичних трендів або сезонних коливань, або що вони були визначені та вилучені.

Процес експоненціально зваженого ковзного середнього заданого часового ряду визначається як:

$$\hat{y_t} = \alpha y_t + (1 - \alpha) y_{t-1},$$

де α - зменшення зважування, постійний коефіцієнт згладжування від 0 до 1,

y_t - це значення в період часу t,

$\hat{y_t}$ - значення EMA прогнозу в будь-який період часу t.

Використовуючи послідовно формулу експоненційно зваженого середнього до всього ряду, можна записати наступним чином:

$$\hat{y}_t = \alpha(y_t + (1-\alpha)y_{t-1} + (1-\alpha)^2 y_{t-2} + ... + (1-\alpha)^k y_{t-k}) + (1-\alpha)^{k+1} y_{t-(k+1)}$$

для всіх $k \in \{0,1,2,..\}$.

Вага кожного спостереження y_{t-i} визначається як $\alpha(1-\alpha)^i$. З плином часу зваженим середнє оцінює все більшу кількість спостережень, а відповідні ваги зменшуються в геометричній прогресії.

У експоненціальному згладжуванні, однак, є один або більше параметрів згладжування, які потрібно визначити, і цей вибір впливає на вибір вагів для зважування спостережень.

Коефіцієнт α набуває значення від 0 до 1. Якщо α близько до 1, то згладжування мале, і y_t приблизно дорівнює $\hat{y}_t$. Це прийнятно у тому випадку, якщо очікуються дуже великі зміни в значенні середнього. У випадку, коли значення α близько до 0, прогноз дає дуже згладжені оцінки середнього значення, і не враховує останні спостереження.

2.2.4 Подвійне експоненціальне ковзне середнє

Метод простого експоненціального згладжування не спрацьовує добре, коли в даних наявний чітко виражений тренд, наприклад різко зростаюча ціна біткоїна. У таких ситуаціях було розроблено кілька методів під назвою «подвійне експоненціальне згладжування» або «експоненціальне згладжування другого порядку», що є рекурсивним застосуванням експоненційного фільтра двічі. Основна ідея подвійного експоненціального згладжування полягає в тому, щоб ввести термін, що враховує можливість ряду мати тренд. Кожен набір даних

часових рядів може бути розкладений на його складові, які є трендом tr_t, сезонною компонентою s_t та нерегулярними коливаннями r_t.

Для того, щоб висловити це в математичній нотації, тепер потрібні три рівняння:

$$a_t = \alpha y_t + (1 - \alpha)(a_{t-1} + b_{t-1})$$
$$b_t = \beta(a_t - a_{t-1}) + (1 - \beta)b_{t-1}$$
$$\hat{y}_{t+h} = a_t + hb_t,$$

де $0 < \alpha < 1$ - коефіцієнт згладжування даних,

$0 < \beta < 1$ - коефіцієнт згладжування тренду.

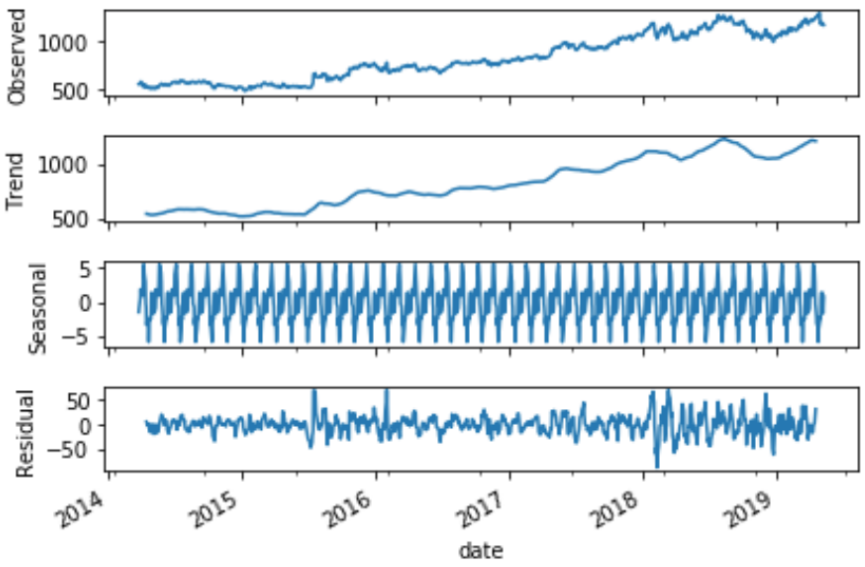


Рисунок 2.1 – Декомпозиція вихідного часового ряду

2.2.5 Потрійне експоненціальне ковзне середнє

Потрійне експоненціальне згладжування, також відоме як метод Голта-Вінтерса застосовує експоненціальне згладжування три рази. Модель використовується для даних, в яких наявні як тренд, так і сезонність.

Вона задається трьома параметрами, які згладжують модель, та має додаткове рівняння для коригування сезонної компоненти.

$$a_t = \alpha(y_t - s_{t-L}) + (1-\alpha)(a_{t-1} + b_{t-1})$$
$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$
$$s_t = \gamma(y_t - a_t) + (1-\gamma)s_{t-L}$$
$$\hat{y}_{t+h} = a_t + hb_t + s_{t+1+(h-1) \bmod L},$$

де $0 < \alpha < 1$ - коефіцієнт згладжування даних,

$0 < \beta < 1$ - коефіцієнт згладжування тренду,

$0 < \gamma < 1$ - коефіцієнт згладжування сезонної компоненти,

L - довжина сезонного циклу,

h - кількість кроків прогнозування[3].

2.3 Методи авторегресії

Модель авторегресії є ефективним інструментом для розуміння і прогнозування майбутніх значень часового ряду, яка включає в себе регресування змінної по значеннях ряду у минулому. Важливість моделей ARMA полягає в їх гнучкості, а також у їхній здатності описувати майже всі особливості стаціонарних часових рядів. Авторегресивні частини цих моделей описують, як послідовні спостереження в часі впливають один на одного, тоді як

частини ковзних середніх захоплюють деякі можливі неспостережувані потрясіння, що дозволяє моделювати різні явища, які можна спостерігати в різних областях від біології до фінансів.

2.3.1 Проста модель авторегресії (AR - Autoregression)

Модель авторегресії вказує, що значення вихідної змінна залежить лінійно тільки від власних попередніх значень і від випадкового шуму.

Позначення *AR(p)* позначає авторегресійну модель порядку *p*. *AR(p)* модель математично визначається як:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + e_n ,$$

де a_i - параметри моделі,

e_n - білий шум.

Найпростішим процесом *AR* є *AR(0)*, який не має залежності між компонентами. На значення прогнозу впливає тільки шум або похибка.
Для *AR(1)* з коефіцієнтом a_1, тільки попереднє значення часового ряду впливає на значення прогнозу.

У випадку якщо a_1 близький до 0, то процес буде виглядати як білий шум, але якщо a_1 за модулем буде наближатися до 1, то вихідне значення буде отримувати більший внесок від попередніх значень ряду порівняно з шумом[3].

2.3.2 Модель авторегресії — ковзного середнього (ARMA - Autoregressive moving average)

Модель ARMA характеризує стохастичний процес за допомогою двох компонентів - авторегресії (AR) та ковзного середнього (MA).

Частина AR передбачає регресування змінної на власні минулі значення. Частина MA включає моделювання похибки як лінійної комбінації похибок, що відбуваються в минулому.

Позначення *ARMA(p, q)* характеризує модель з p авторегресійними компонентами і q компонентами для ковзного середнього:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + \sum_{i=1}^{q} b_i e_{n-i} + e_n ,$$

де a_i, b_i - параметри моделі,

e_n - білий шум.

Модель ARMA є доречною, коли система є функцією ряду неспостережуваних потрясінь та власної поведінки. Наприклад, ціни на акції можуть бути суттєво змінитись за якоїсь фундаментальної інформації, а також продемонструвати ефекту повернення до середнього через певні дії учасників ринку.

2.3.3 Модель авторегресії — інтегрованого ковзного середнього (ARIMA - Autoregressive integrated moving average)

Модель ARIMA є ускладненням моделі ARMA. Вона складається з трьох компонентів:

1. авторегресії (AR), що використовує залежний зв'язок між спостереженням та іншими спостереженнями із затримкою в часі.
2. інтегрована компонента (I), що диференціює вхідні спостереження з метою зробити часовий ряд стаціонарним.
3. ковзного середнього (MA), що використовує залежність між спостереженням і відхиленням від ковзного середнього попередніх спостережень.

Кожна з цих компонентів задає параметри моделі. Використовується стандартне позначення *ARIMA(p,d,q)*, де параметри позначають:

1. *p* - кількість лагових спостережень, включених в модель;
2. *d* - кількість разів, коли до вхідних спостережень було застосовано диференціювання. Параметр також називається ступенем диференціювання;
3. *q* - розмір вікна ковзного середнього, або порядок ковзного середнього.

Для застосування моделі ARIMA використовуються формули моделі ARMA, проте на вхід замість y_t подається $\Delta y_t = y_t - y_{y-1}$.

Для кожного параметра можна використовувати значення 0, яке вказує на відсутність відповідного компоненту моделі. Тобто модель ARIMA може бути налаштована для виконання функцій моделі ARMA чи навіть простий AR, I або MA.

Слід також зауважити, що AR і MA - це лінійні моделі, які працюють на стаціонарних часових рядах, в той час як модель I є процедурою попередньої обробки даних, щоб звести ряд до стаціонарного, якщо це необхідно.

2.3.4 Знаходження значень параметрів

Відповідні значення p і q можуть бути знайдено за допомогою побудови часткових автокореляційних функцій для оцінки p, а також з використанням автокореляційних функцій для оцінки q. Для цього використовують інформаційний критерій Акайке (Akaike information criterion - AIC). При оцінці кількості інформації, втраченої моделлю, цей критерій шукає компроміс між точністю та простотою моделі, тому моделі, які краще підходять при використанні меншої кількості функцій, отримають кращу (нижчу) оцінку AIC, ніж аналогічні моделі, які використовують більше можливостей.

В загальному вигляді критерій визначається як:

$$AIC = 2k - 2ln(\hat{L})$$

де k - кількість оцінених параметрів у моделі,

$\hat{L}$ - максимальне значення функції правдоподібності для моделі.

Модель із мінімальним значенням AIC вважається найкращою. Таким чином, AIC сприяє точності моделі (оцінюючи функцію правдоподібності), але також “штрафує” за збільшення кількості оцінюваних параметрів. Такий штраф перешкоджає перенавчанню моделі, оскільки збільшення кількості параметрів у моделі майже завжди покращує точність відтворення моделі на тестових даних. У сучасному середовищі для аналіза та прогнозування, цей критерій реалізовано у відповідних бібліотеках та функціях.

Висновки до розділу 2

У розділі 2 було розглянуто традиційні методи прогнозування вартості акцій, а саме методи згладжування та методи авторегресії. У пункті 2.2 та 2.3 було наведено відповідні теоретичні відомості методів, які було реалізовано в дослідженні.

Також було розглянуто параметри, які впливають на побудову прогнозу кожним методом.

РОЗДІЛ 3 МАТЕМАТИЧНІ МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ ПРОГНОЗУВАННЯ ВАРТОСТІ АКЦІЙ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

3.1 Вступ до розділу 3

Штучні нейронні мережі (Artificial Neural Networks - ANN) мають перевагу в прогнозуванні часових рядів, оскільки мають потенціал для вирішення складних проблем прогнозування.

Важлива особливість ANN стосовно застосування до проблем прогнозування часових рядів полягає в здатності нейронних мереж до нелінійного моделювання, без будь-якого припущення про статистичний розподіл часового ряду. Кожна модель адаптивно формується на основі даних. З цієї причини штучні нейронні мережі керуються даними та є самоадаптивними за своєю природою[4].

У цьому розділі буде розглянуто використання нейронних мереж на фінансових ринках, їх вхідних та вихідних даних, та архітектури мережі.

3.2 Загальні відомості про нейронні мережі

Загальна структура штучної нейронної мережі ґрунтується на сукупності з'єднаних вузлів - нейронів (подібно до біологічних нейронів у головному мозку людини).

Модель ANN може мати три типа шарів - вхідний, прихований і вихідний шар, з'єднані ациклічними зв'язками. Моделі також можуть мати більше одного прихованого шару. В якості приклада тришарова архітектура найпростішої ANN схематично зображується так (рис. 3.1):

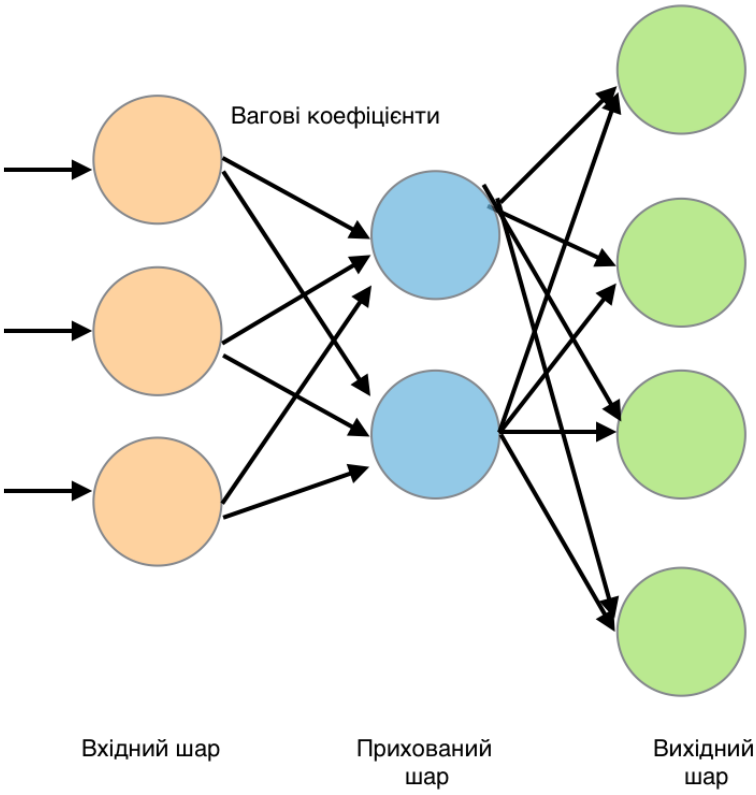


Рисунок 3.1 – Схема найпростішої нейронної мережі

Вихідне значення нейронної мережі математично задається так:

$$y_t = \alpha_0 + \sum_{j=1}^{q} \alpha_j g(\beta_{0j} + \sum_{i=1}^{p} \beta_{ij} y_{t-i}) + \varepsilon_t, \ \forall t \,,$$

де p – кількість вхідних змінних,

q – кількість прихованих вузлів,

α_j та β_{ij} – вагові коефіцієнти,

ε_t – випадковий шум.

В якості функції g можна використовувати наступні функції:

1. сигмоїдна функція;

$$f(x) = \frac{1}{1+e^{-x}} .$$

В деякій літературі ця функція зветься логістичною. Ця нелінійна функція є одною з найпоширених функцій активацій для глибинного навчання.

2. гіперболічний тангенс;

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Ця функція дає кращий результат порівняно з сигмоїдною функцією для багатошарових нейронних мереж. Однак функція не вирішує проблему зникаючого градієнта. Головною перевагою, яку надає функція, є те, що вона центрована відносно нуля, що допомагає в процесі зворотного поширення помилки.

3. softmax функція;

$$f(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Функція Softmax є ще одним типом функції активації, що використовується в нейронних обчисленнях. Вона використовується для обчислення ймовірності розподілу вектора дійсних чисел, та набуває значення діапазоном від 0 до 1.

4. ReLU функція;

$$f(x) = max(0, x)$$

Ця функція вважається найбільш успішною і широко використовуваною передавальною функцією. ReLU показує кращу продуктивність в глибокому навчанні порівняно з сігмоїдою та гіперболічним тангенсом. ReLU являє собою майже лінійну функцію і тому зберігає властивості лінійних моделей, які роблять їх легко оптимізувати методами градієнтного спуску. Основна перевага використання ReLU в обчисленні полягає в тому, що функція не потребує обчислення експоненти чи ділення, отже гарантує більш швидке виконання[5].

Загалом, існує безліч варіацій кожної передавальної функції, проте в даній роботі будемо розглядати найбільш поширені, що було наведено вище.

Безпосереднє навчання відбувається за рахунок мінімізації функції похибки при реалізації методу зворотного поширення помилки. Цільовою функцією є середньоквадратична функція помилки (функція втрат / вартість). Ми повинні знайти оптимізаційні значення ваг нейронної мережі, щоб мінімізувати цільову функцію.

До найбільш поширених методів оптимізації відносяться:

1. метод градієнтного спуску;

$$x_{t+1} = x_t - \alpha \cdot f'(x_t)$$

Згідно з методом, задля мінімізації функції здійснюються кроки, пропорційні протилежному значенню градієнту. Параметром цього методу є швидкість спуску α. При великому значенні α буде можливо робити більші

кроки для знаходження мінімуму, але існує ризик перескочити найнижчу точку. При дуже низькій швидкості навчання алгоритм буде впевнено рухатися в напрямку негативного градієнта, але реалізація в такому випадку займе багато часу.

2. метод стохастичного градієнтного спуску;

Це тип градієнтного спуску, який обробляє 1 елемент для навчання на кожну ітерацію. Отже, параметри оновлюються навіть після однієї ітерації, в якій оброблено лише одне значення змінної. Це дає змогу оптимізувати цільову функцію набагато швидше, ніж звичайний градієнтний спуск. Але якщо кількість даних для навчання дуже велика, кількість ітерацій буде відповідно досить великою.

3. метод градієнтного спуску міні-серіями;

Це тип спуску з градієнтом, який працює швидше, ніж серійний та стохастичний градієнтний спуск. Нехай існує m кількість вхідних даних, тоді за одну ітерацію такого методу буде оброблятись $b < m$ елементів. Отже, навіть якщо кількість навчальних даних велика, вона обробляється в менших групах тренувань на один раз. Таким чином, метод працює для великої кількість даних для навчання, оптимізуючи цільову функцію з меншою кількістю ітерацій.

4. градієнтний спуск з моментом;

Градієнтний спуск з моментом - це метод, який допомагає прискорити спуск у відповідному напрямку і гасить коливання, наближаючись до локального мінімума. Це реалізується за допомогою додавання до вектора поточного оновлення частку вектора оновлення з попереднього кроку:

$$v_t = \gamma v_{t-1} + (1 - \beta) \cdot dW(x_t)$$

$$W = W - \alpha v_t ,$$

де v і dW аналогічні прискоренням і швидкості,

α є швидкістю навчання,

β зазвичай зберігається при 0,9.

5. метод Адама (Adaptive Moment Estimation)

Оптимізатор Адама є одним з найпопулярніших алгоритмів оптимізації спуску градієнтів, оскільки є обчислювально ефективним і має дуже мало вимог до пам'яті. Цей метод обчислює індивідуальну адаптивну швидкість навчання для кожного параметра з оцінок першого і другого моментів градієнтів.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_1) g_t{}^2 ,$$

де m_t та v_t - оцінки першого та другого моменту відповідно.

Параметри оновлюються за наступним законом:

$$\hat{m_t} = \frac{m_t}{1-\beta_1{}^t}$$
$$\hat{v_t} = \frac{v_t}{1-\beta_2{}^t}$$
$$\theta_{t+1} = \theta_t - \alpha \frac{m_t}{\sqrt{v_t}+\varepsilon} .$$

3.3. Вхідні дані

Одним з найважливіших факторів для побудови нейронної мережі є визначення того, на яких даних буде тренуватись мережа. В цій роботі мета більшості цих мереж - якомога точніше спрогнозувати ціну акції, щоб вирішити, коли купувати або продавати цінні папери на основі попередніх даних ринку. Завдання полягає у визначенні того, які індикатори та вхідні дані будуть використані, та у зборі достатньої кількості навчальних даних для відповідного навчання системи.

Вхідними даними можуть бути відомості про обсяг, ціну або щоденну зміну в ціні, але ANN можуть також отримувати на вхід дані, такі як технічні індикатори (ковзні середні, показники трендової лінії тощо) або фундаментальні показники (внутрішня вартість акцій, показники економічного середовища тощо).

Корисним прикладом нейронної мережі є JSE-system[6], що моделювала поведінку Йоганнесбургзької фондової біржі. Ця система мала 63 показники з різних категорій для того, щоб отримати загальне уявлення про ринкове середовище. Значення цих 63 вхідних індикаторів можна розділити на наступні класи:

1. фундаментальні - обсяг продажів, прибутковість, ціна;
2. технічні - ковзні середні, тренди обсягу продаж тощо;
3. Індекси JSE - ринкові індекси для різних секторів: золото, метали;
4. міжнародні індекси - промисловий індекс Доу-Джонса та ін.;
5. ціна на золото / курси валют;
6. процентні ставки;
7. економічні показники - загальний експорт, імпорт.

Хоча остаточна ANN JSE-system приймала на вхід усі 63 входами, аналіз показав, що багато з них виявились надлишковими. В процесі тренування автори цієї моделі відкинули 20 змінних, зменшив кількість прихованих вузлів від 21 до 14, з незначним впливом на продуктивність системи, а також зменшивши час навчання.

Інші системи нейронних мереж можуть використовують інші типи вхідних даних. Простіші системи можуть використовувати лише минулі ціни на акції. Система, розроблена Юном[7], побудована на вхідний даних частотах ключових фраз, використаних у звіті президента перед акціонерами. В якості вхідних даних для прогнозування вартості акцій дослідники Валчанського технологічного інституту Джейбхай, Аргідді та Апте поєднали цифрові дані та інтелектуальний аналіз тексту новин[8].

Визначення належних вхідних даних є першим кроком у навчанні мережі. Другий крок - представлення вхідних даних таким чином, що дозволяє мережі правильно вчитися без перетренування. В даній роботі для того, щоб дати більш об’єктивну порівняльну характеристику для класичних методів та методів машинного навчання, за вхідні дані було обрано попередні значення вартості акцій.

3.4 Багатошаровий перцептрон Румельхарта (Multilayer Perceptron - MLP)

Штучні нейронні мережі виглядають як електронні моделі, засновані на нейронній структурі мозку, яка в основному навчається з досвіду. Найпростішим видом нейронної мережі є одношарова персептронная мережа (в загальному випадку це є багатошаровий перцептрон Румельхарта), яка складається з одного шару вихідних вузлів, а входи подаються безпосередньо на виходи через ряд ваг.

Таким чином, його можна вважати найпростішою формою мережі передачі інформації. Багатошарова нейронна мережа складається з декількох шарів обчислювальних одиниць, зазвичай взаємопов'язаних між собою в прямому напрямку. Кожен нейрон в одному шарі спрямовував з'єднання до нейронів наступного шару. У багатьох додатках одиниці цих мереж застосовують сигмовидну функцію як функцію активації. Універсальна теорема апроксимації для нейронних мереж стверджує, що кожну безперервну функцію, яка відображає інтервали дійсних чисел до деякого вихідного інтервалу дійсних чисел, можна апроксимувати довільно тісно багатошаровим персептроном лише одним прихованим шаром. Цей результат виконується для широкого діапазону функцій активації, як у випадку сигмоїдних функцій[9].

3.4 Згорткова нейронна мережа (Convolutional Neural Network - CNN)

Згорткова нейронна мережа - це тип штучної нейронної мережі, в якій картина зв'язності між її нейронами натхненна організацією зорової кори тварини, окремі нейрони якої розташовані таким чином, що вони реагують на перекриваються області поля. Існує кілька різних теорій про те, як точно визначити таку модель, але всі різні реалізації можуть бути реалізовані наступними кроками:

1. реалізація згортки кожним фільтром для формування карти збудження;
2. максимізаційне агрегування області інтересу, отриманої з першого кроку;
3. повторення кроків 1 і 2, поки не залишуться ознаки достатньо високого рівня;
4. використання MLP для вирішення конкретної задачі, використовуючи результати попередніх кроків як вхідні дані.

Головною перевагою згорткових нейронних мереж є те, що ми використовуємо згорткові шари, щоб виявити ознаки мережі, що дозволяє тренувати нейронну мережу без складної попередньої обробки, оскільки корисні функції будуть вивчені під час навчання[10]. У нашому випадку, замість того, щоб розглядати стандартний підхід CNN для двовимірних зображень, в роботі буде застосовано згорткову мережу до 1D даних, а саме до часових рядів вартості акцій компанії.

Графічно схему згорткової нейронної мережі можна представити так (рис. 3.2):

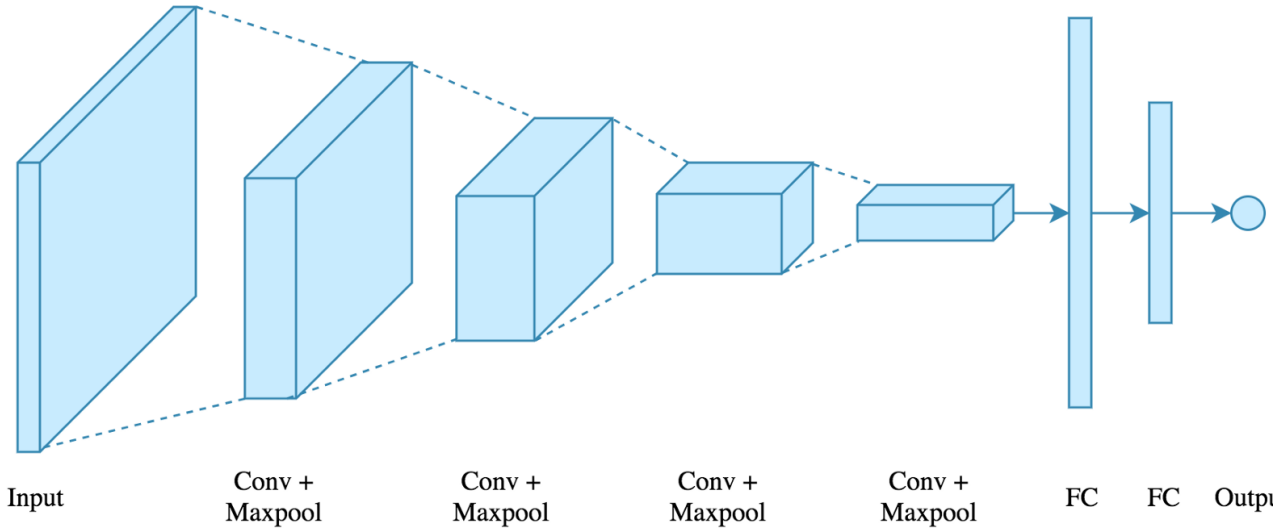


Рисунок 3.2 – Схема згорткової нейронної мережі

3.5 Рекурентна нейронна мережа (Recurrent Neural Network - RNN)

Рекурентна нейронна мережа - це будь-яка штучна нейронна мережа, нейрони якої передають сигнали зворотного зв'язку один одному. Ідея RNN полягає у використанні послідовної інформації. У традиційній нейронній мережі ми припускаємо, що всі входи (і виходи) незалежні один від одного. Але для

багатьох завдань це не найкраща ідея. Зокрема, якщо хочеться передбачити наступне слово в реченні, то краще, якщо відомо, які слова прийшли до нього. RNN називаються рекурентними нейронними мережами, тому що вони виконують одне і те ж завдання для кожного елемента послідовності, при цьому вихідні дані залежать від попередніх обчислень[11]. Варто відзначити, що теоретично RNN може враховувати як завгодно віддалені в минулому послідовності інформації, але на практиці вони обмежуються оглядом лише на кілька кроків назад. Цей тип нейронних мереж цілком підходить для прогнозування вартості акцій, оскільки майбутні кроки можуть залежати від минулих.

Висновки до розділу 3

Останні дослідження використання нейронних мереж в задачах прогнозування доводять доцільність та перспективність використання цих методів, В розділі 3 було розглянуто математичні основи нейронних мереж, а саме багатошарового перцептрону Румельхарта, згорткових нейронних мереж та рекурентних нейронних мереж.

Суттєвим є правильно обрані вхідні дані. В даній роботі в якості вхідних даних брались тільки значення ціни акції в минулому. Проте в розділі 3 також зазначені інші підходи на приклади відомих дослідницьких робіт в цій сфері.

РОЗДІЛ 4 РЕЗУЛЬТАТИ РЕАЛІЗАЦІЇ МЕТОДІВ ПРОГНОЗУВАННЯ НА ДАНИХ

4.1 Вступ до розділу 4

У цьому розділі буде наведено обчислювальні результати, пов'язані з навчальними процесами, а також детальний опис реалізації дослідження та архітектури програмного продукту.

В якості мови програмування була обрана мова Python 3.0. Перевагами саме такого вибору стали наявність широкого вибору бібліотек та фреймворків для роботи з даними, стислість та читабельність коду (що особливо доцільно для задач машинного навчання).

Для реалізації методів авторегресії та згладжування було застосовано бібліотеку statsmodels. Це бібліотека мови Python, який дозволяє користувачам досліджувати дані, оцінювати статистичні моделі і виконувати статистичні тести. Statsmodels побудована поверх чисельних бібліотек NumPy і SciPy та інтегрується з бібліотекою Pandas для обробки даних.

Всі штучні нейронні мережі були реалізовані з використанням бібліотеки методів глибинного навчання Keras. Кожна побудована ANN потребувала для навчання від 100 до 200 епох.

4.2 Реалізація методів згладжування

В даній роботі було розглянуто 5 методів згладжування, кожен з яких було реалізовано для декількох параметрів. Кожен наступний метод являє собою ускладнення попереднього, отже вочевидь дає точніші результати.

Метод ковзного середнього було реалізовано для різних значень параметру N - кількості попередніх моментів часу, що було взяти до уваги при побудові прогнозу (рис. 4.1). Було перевірено, що при зменшенні довжини вікна N модель показує більш точний результат на тестовій вибірці, що вказує на властивість найостанніших даних мати вплив на прогнозну оцінку в майбутньому.

Рисунок 4.1 – Графік прогнозу простого ковзного середнього

Метод зваженого середнього було реалізовано за допомогою методу ewma бібліотеки Pandas (рис. 4.2). За документацією відповідні вагові коефіцієнти обчислювались за наступною формулою:

$$\alpha = 1 - e^{\frac{log(0.5)}{halflife}}.$$

Тобто параметром налаштування моделі стало $halflife$. Це дало наступні результати для різних значень $halflife$:

Рисунок 4.2 – Графік прогнозу методом зваженого ковзного середнього

Аналогічний підхід використовується для експоненційного ковзного середнього (рис. 4.3), де як параметр задається рівень згладжування - коефіцієнт α, що являє собою ступінь зменшення зважування від 0 до 1.

Рисунок 4.3 – Графік прогнозу методом експоненціального ковзного середнього

Чим менший рівень згладжування, тим точніший є прогноз, оскільки кожне попереднє значення важить більше.

Метод подвійного експоненціального згладжування передбачав задання додатково параметру β, який відповідає за згладжування тренду (рис. 4.4). Комбінація пари α та β коригувало точність та якість прогнозу.

Рисунок 4.4 – Графік прогнозу методом подвійного експоненціального ковзного середнього

Як було зазначено в розділі 2 потрійне експоненціальне ковзне середнє або метод Голта-Вінтерса використовується для даних, в яких наявні як тренд, так і сезонність, та має додатковий коефіцієнт згладжування сезонної компоненти γ . В роботі розглядається саме адитивна модель тренду та сезонної компоненти часового ряду. Функція ExponentialSmoothing в модулі tsa.holtwinters дозволяє задавати модель лише кількістю сезонних періодів, що наявні в даних. Отриманий результат відображено у наступному малюнку (рис 4.5).

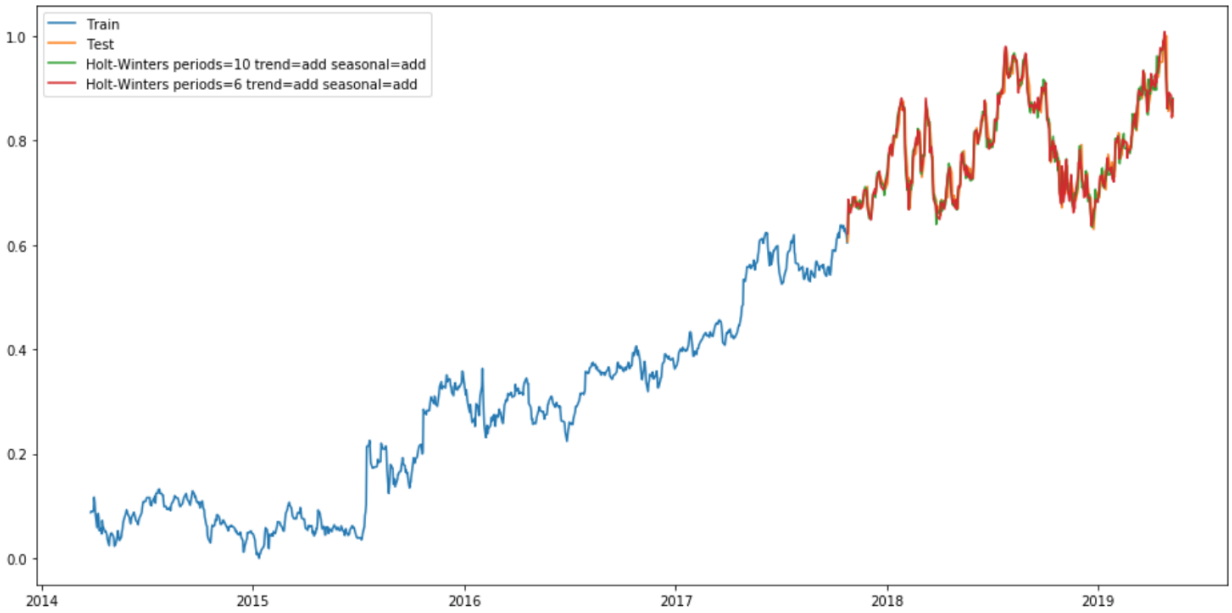


Рисунок 4.5 – Графік прогнозу методом Голта-Вінтерса

Отже, в даному випадку зменшення кількості періодів приводило до більш точного прогнозу вартості акцій.

Результати реалізації усіх методів згладжування наведено в таблиці 1.

Таблиця 1 – Результати реалізації методів згладжування

Метод	Характерні параметри	Test MSE	Test MAE
Ковзне середнє	Вікно $N = 30$	0.0029	0.0455
	Вікно $N = 10$	0.0012	0.0281
	Вікно $N = 5$	0.0005	0.0180
Зважене ковзне середнє	*halflife* = 12	0.0025	0.0419
	halflife = 1	0.0001	0.0089

Продовження таблиці 1

	halflife = 0.5	0.000026	0.0037
Експоненціальне ковзне середнє	Рівень згладжування α = 0.1	0.0018	0.0340
	Рівень згладжування α = 0.2	0.0011	0.0259
	Рівень згладжування α = 0.6	0.00049	0.0165
Подвійне експоненціальне ковзне середнє	α = 0.1, β = 0.3	0.00244	0.0388
	α = 0.2, β = 0.8	0.00163	0.0322
	α = 0.6, β = 0.6	0.00054	0.0177
Потрійне експоненціальне ковзне середнє	*period* = 10	0.00045	0.0162
	period = 6	0.00042	0.0154

4.3 Реалізація методів авторегресії

Як було зазначено в розділі 2, в роботі розглядається 3 класичних методи авторегресії. Кожен наступний метод показував кращі результати та мав більш складне математичне підґрунтя. Для цих методів коефіцієнт Акайке дозволив програмно обрати модель, що дає найкращий прогноз. Отже, у випадку даного датасету це виявилась модель ARIMA(2,1,1).

Результати виконання усіх трьох методів зведені в одну порівняльну таблицю 2.

Таблиця 2 – Результати реалізації методів авторегресії

Метод	Характерні параметри	Test MSE	Test MAE
Проста модель авторегресії	$p = 2$	0.000347	0.013322
Модель авторегресії — ковзного середнього	$p = 3$, $q = 1$	0.000348	0.013346
Модель авторегресії — інтегрованого ковзного середнього	$p = 2$, $q = 1$, $d = 1$	0.000345	0.013335

4.4 Реалізація MLP

Основною характеристикою багатошарового перцептрону є його архітектура, а саме кількість прихованих шарів, вузлів та активуючих функцій. Також, оскільки результат навчання частково залежить від ініціалізації змінних, кожна модель тренувалась окремо 5 разів, та всі характеристики для порівняльної таблиці є усередненими значеннями. Для більш глибокого дослідження моделі в роботі було реалізовано 10 архітектур, ефективність кожної з яких наведено в таблиці 3:

Таблиця 3 – Результати навчання MLP

#	Модель	Кількість параметрів	Training MAE	Training MSE	Test MAE	Test MSE	Алгоритм	Функція активації
1	MLP 20-10-1	221	0.0645	0.0074	0.090	0.014	adam	relu
2	MLP 20-20-1	441	0.0462	0.0038	0.052	0.005	adam	relu
3	MLP 20-100-1	2201	0.0438	0.0038	0.037	0.002	adam	relu
4	MLP 20-500-1	11001	0.0398	0.0032	0.036	0.002	adam	relu
5	MLP 30-500-1	16001	0.0387	0.0029	0.035	0.002	adam	relu
6	MLP 10-500-1	6001	0.0354	0.0025	0.034	0.002	adam	relu
7	MLP 20-100-1	2201	0.0035	0.0428	0.032	0.002	adam	tanh
8	MLP 20-100-1	2201	0.0593	0.0064	0.046	0.004	sgd	tanh

Продовження таблиці 3

9	MLP 20-100-1	2201	0.0668	0.0078	0.093	0.014	sgd	relu
10	MLP 20-100-1	2201	0.0572	0.0064	0.059	0.005	rmsprop	relu

За результатами цієї порівняльної таблиці цілком простежується залежність якості прогнозу від кількості вузлів у прихованому шарі, методу оптимізації та функції активації. Найкращий результат показала модель MLP 20-100-1 с методом оптимізації Adam, та гіперболічним тангенсом в ролі активуючої функції. Наведемо графік вартості акції, передбаченої цією моделлю (рис 4.6), а також графік похибки прогнозу (рис. 4.7).

Рисунок 4.6 – Графік прогнозу MLP-20-100-1

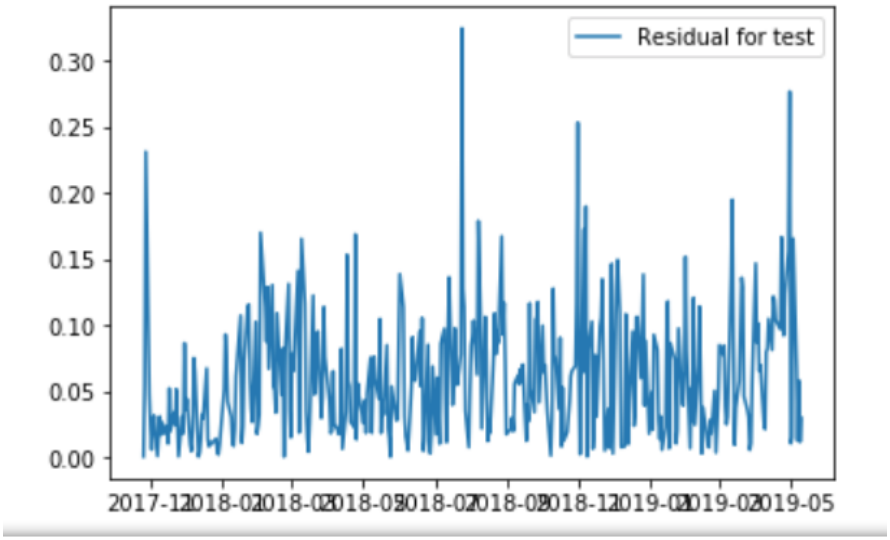


Рисунок 4.7 – Графік похибки MLP-20-100-1

4.5 Реалізація згорткових нейронних мереж (CNN)

На відміну від багатошарового перцептрони, згорткові нейронні мережі мають складнішу структуру, оскільки вимагають калібрувати кількість та тип шарів, кількість вузлів в кожному шарі, метод оптимізації, функцію активації та кількість фільтрів. Отже позначення CNN-20-200-3 буде позначати 20 вузлів на вхід як перший шар, 200 фільтрів розміром 3x3, MaxPooling шар та 2 fully-connected шарів. В даній роботі було протестовано 7 різних архітектур CNN, які показали наступні результати таблиці 4:

Таблиця 4 – Результати навчання CNN

#	Модель	Кількість параметрів	Training MAE	Training MSE	Test MAE	Test MSE	Алгоритм	Функція активації

Продовження таблиці 4

1	CNN 20-256-3	199,937	0.0212	9.4573e-04	0.1243	0.0216	adam	relu
2	CNN 20-256-5	201,367	0.0182	7.4266e-04	0.174	0.0388	adam	relu
3	CNN 10-256-3	198,657	0.0238	0.0011	0.096	0.0137	adam	relu
4	CNN 20-256-3	199,937	0.0513	0.0050	0.29	0.110	sgd	relu
5	CNN 20-500-3	756,501	0.0515	0.0050	0.2592	0.0890	sgd	relu
6	CNN 20-300-3	273,901	0.0342	0.0021	0.121	0.0282	prmsprop	relu
7	CNN 20-300-3	273,901	0.0414	0.0030	0.255	0.079	prmsprop	tanh

Найкращий результат показала модель CNN 20-256-3 с методом оптимізації Adam, та активуючій функцією ReLU. Наведемо графік вартості акції, передбаченої цією моделлю (рис. 4.8).

Рисунок 4.8 – Графік прогнозу CNN 20-256-3

4.5 Реалізація рекурентних нейронних мереж (RNN)

Для реалізації прогнозування рекурентних нейронних мереж було обрано модель LSTM (Long Short-Term Memory Units).

LSTM допомагають зберегти помилку, яку можна розповсюджувати через час і шари. Підтримуючи більш постійну помилку, вони дозволяють повторним мережам продовжувати вивчати протягом багатьох кроків часу[11]. LSTM містять інформацію за межами нормального потоку повторюваної мережі в закритій комірці. Інформацію можна зберігати, записувати або читати з клітини, подібно до даних у пам'яті комп'ютера. Клітинка приймає рішення про те, що зберігати, і коли дозволити читання, запис і стирання, через ворота, які

відкриваються і закриваються. На відміну від цифрового зберігання на комп'ютерах, ці гейти є аналоговими, реалізованими з елементарним множенням на сигмоїди, які все знаходяться в діапазоні 0-1. Ці ворота діють на сигнали, які вони отримують, і подібно до вузлів нейронної мережі, вони блокують або передають інформацію на основі своєї сили та імпорту, яку вони фільтрують своїми власними наборами ваг. Тобто, клітини вивчають, коли дозволяють вводити дані, залишати їх або видаляти через ітераційний процес внесення припущень, помилок, що поширюються назад, і коригування ваг за допомогою градієнтного спуску[12].

Для аналізу параметрів впливу на прогноз вартості акцій було реалізовано 5 варіантів архітектуру LSTM, які було зведено в наведену нижче таблицю 4. Позначення LSTM-20-30-30-30-1 використовувалось для визначення нейронної мережі LSTM, з початковими даними за 20 днів назад, кількістю вузлів 30 на першому, другому та третьому шарі LSTM та одним fully-connected шаром з отриманим виходом з одного елементу.

Таблиця 5 – Результати навчання LSTM

#	Модель	Кількість параметрів	Training MAE	Training MSE	Test MAE	Test MSE	Алгоритм	Функція активації
1	LSTM 20-30-30-30-1	18,631	0.0074	1.0564e-04	0.0198	0.0005	adam	relu

Продовження таблиці 5

2	LSTM 20-30-30-40-1	22,681	0.0079	1.8716e-04	0.0188	0.0005	adam	relu
3	LSTM 30-30-30-30-1	18,631	0.0106	1.9467e-04	0.0159	0.0004	adam	relu
4	LSTM 10-30-30-30-1	18,631	0.0099	1.7086e-04	0.0150	0.0003	adam	relu
5	LSTM 10-30-30-30-1	18,631	0.0082	1.2895e-04	0.01522	0.00036	sgd	relu

Найкращий результат показала модель LSTM 10-30-30-30-1 с методом оптимізації Adam, та активуючею функцією ReLU та з найменшим вікном - лагом. Наведемо графік вартості акції, передбаченої цією моделлю (рис. 4.9).

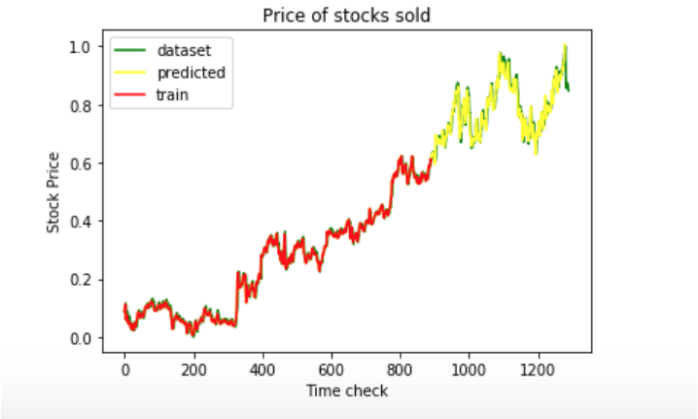


Рисунок 4.9 – Графік прогнозу LSTM 10-30-30-30-1

Висновки до розділу 4

Порівняно з класичними методами, ця работа дала змогу підтвердити перспективність та доцільність використання штучних нейронних мереж для подальшого дослідження їх застосування на фінансових ринках.

Загалом виконання класичних алгоритмів та алгоритмів машинного навчання в розділі 4 можна звести у порівняльну таблицю 6.

Таблиця 6 – Порівняльна таблиця реалізованих методів

Моделі	Переваги	Недоліки
Згладжування	Здатність обробляти тенденції змінних рівнів і компоненти сезонності	Вразливі до екстремальних значень
Авторегресія	Можна легко автоматизувати	Сильні обмеження в припущеннях
Штучні нейронні мережі - ANN	Можливість обробки складних нелінійних шаблонів. Висока точність прогнозу	Потребує велику кількість даних.

РОЗДІЛ 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ

5.1 Постановка завдання проектування

Проводиться оцінка основних характеристик програмного продукту, призначеного для прогнозування часових рядів на прикладі вартості акцій. Моделювання проводиться у середовищі розробки Jupyter Notebook за допомогою мови програмування Python.

Результати моделювання призначені для використання на персональних комп'ютерах під управлінням операційної системи Windows, Linux та Mac OS при наявності встановленого дистрибутиву для мов програмування Python та R з відкритим кодом Anaconda.

Нижче наведено аналіз різних варіантів реалізації продукту з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз — це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода функціонально-вартісного аналізу називаються функціями.

Мета функціонально-вартісного аналізу полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу

функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Такий аналіз дозволяє провести дослідження продукту заради його найоптимальнішої реалізації. Зокрема, виключення надлишкових або неефективних функцій дозволяє зменшити витрати на виробництво.

5.2. Обґрунтування функцій програмного продукту

Головна функція F_0 – використання програмного продукту, який аналізує отримує на вхід дані та будує його модель для прогнозу вартості акцій. Виходячи з конкретної мети, можна виділити наступні основні функції програмного продукту:

1. F_1 – вибір мови програмування;
2. F_2 – вибір бібліотеки для глибинного навчання нейронних мереж;
3. F_3 – візуалізація даних.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Python;

б) мова програмування R;

Функція F_2:

а) бібліотека Keras;

б) бібліотека scikit-learn.

Функція F_3:

а) за допомогою переносу даних в Excel ;

б) програмно за допомогою бібліотеки Matplotlib.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 5.1).

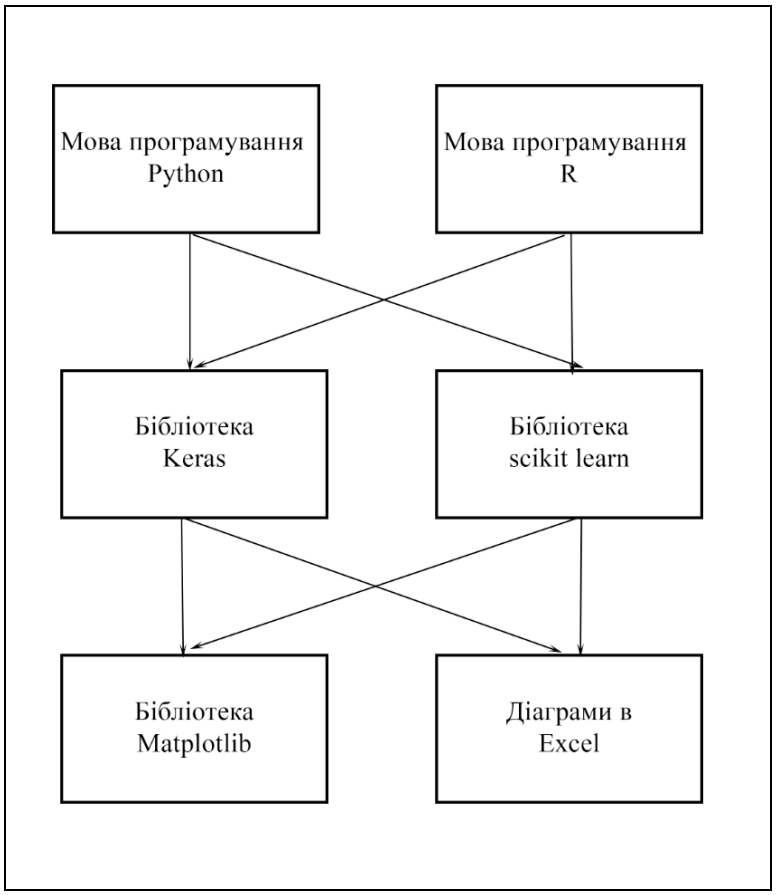


Рисунок 5.1 – Морфологічна карта

На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій, яка представлена в таблиці 7.

Таблиця 7 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки

Продовження таблиці 7

F_1	*А*	Легко побудувати нові моделі з нуля – матричні обчислення та оптимізація бібліотек для обробки даних	Менша кількість бібліотек для машинного навчання
	Б	Наявність великої кількості бібліотек	Займає більше часу при написанні коду
F_2	*А*	Широкий функціонал для глибинного навчання	Складніший у вивченні
	Б	Простий та доступний у використанні навіть для початківців	Виконує лише базові функції
F_3	*А*	Швидкість реалізації	Потребує додаткових навичок кодування
	Б	Зрозумілий інтерфейс	Обмежений функціонал

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

1. Функція F_1

Оскільки розробка програмного продукту потребує написання чіткого та зрозумілого коду, дуже важливим фактоpо є наявність читабельного коду, який мінімізує час написання програми та читання коду. Тому для реаліщації цих вимог доцільно відкинути варіант б).

2. Функція F_2

Оскільки основною задачею продукту є розробка методів прогнозування, який потребують швидку реалізацю фундаментальних математичних операцій, то використання бібліотекз широким функціоналом, які надають ці можливості, є нагальним питанням. В Отже, варіант б) має бути відкинутий.

3. Функція F_3

Візуалізація результатів не відіграє велику роль у даному програмному продукті, оскільки якість малюнків зберігається при обох варіатах, тому вважаємо варіанти а) та б) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації програмного продукту:

$$F_1\text{а} - F_2\text{а} - F_3\text{а}$$
$$F_1\text{а} - F_2\text{а} - F_3\text{б}.$$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.3 Обґрунтування системи параметрів програмного продукту

Маючи вимоги щодо основних функцій, які реалізуються в програмному продукті, визначають основні параметри виробу, які надалі використовуватимуться для розрахунку коефіцієнта технічного рівня.

Введемо наступні параметри:

1. *X1* – точність розв’язку;
2. *X2* – потенційний об’єм програмного коду;
3. *X3* – час обробки даних алгоритмом;
4. X4 – кількість додатково імпортованих бібліотек

X1: Показує точність реалізованого прогнозу, тобто відсоток похибки. *X2:* Показує розмір програмного коду який необхідно створити безпосередньо розробнику. *X3*: Відображає час, який витрачається на виконання програмного коду. *X4*: Показує кількість додатково необхідних бібліотек для реалізації продукту.

Таблиця 8 – Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Точність розв’язку	X1	%	10	5	1
Потенційний об’єм програмного коду	X2	кількість строк коду	1200	800	600

Продовження таблиці 8

Час обробки даних алгоритмом	X3	с	1500	900	300
Кількість додатково імпортованих бібліотек	X4	одиниці	10	7	5

За даними таблиці 8 будуються графічні характеристики параметрів (рис. 5.2 – рис. 5.5).

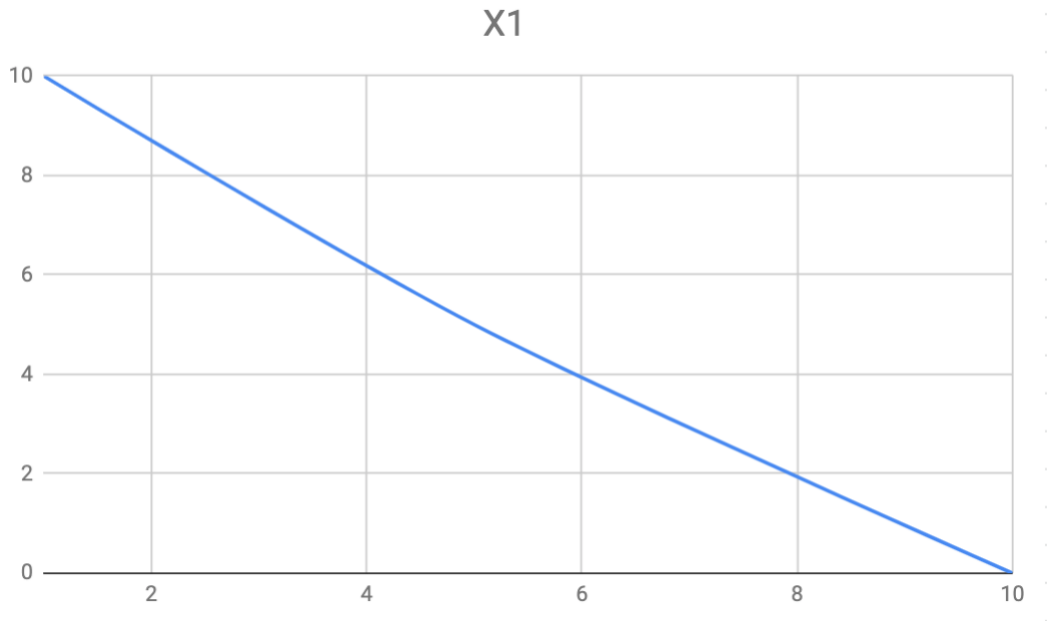


Рисунок 5.2 – X1, точність розв’язку.

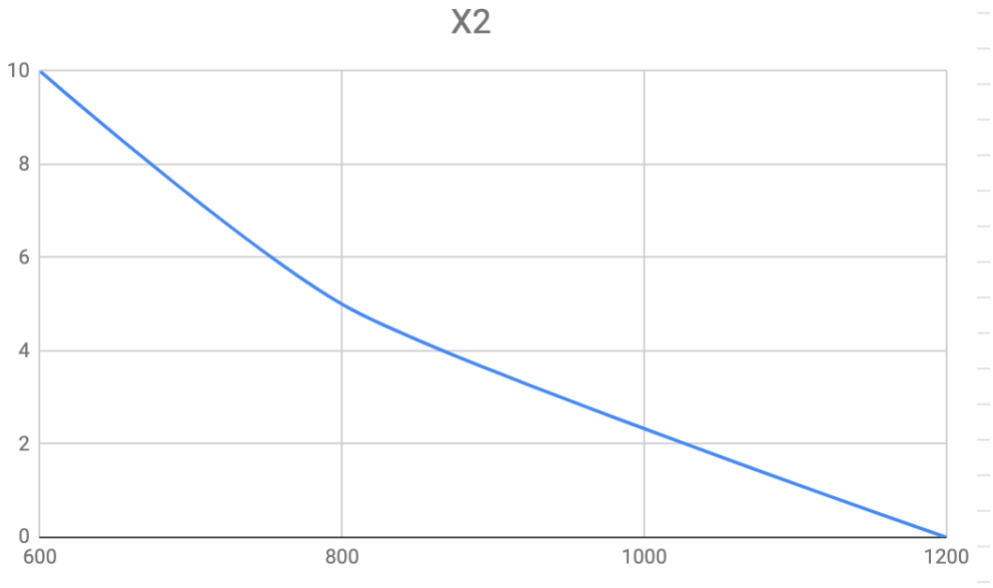


Рисунок 5.3 – X2, потенційний об’єм програмного коду.

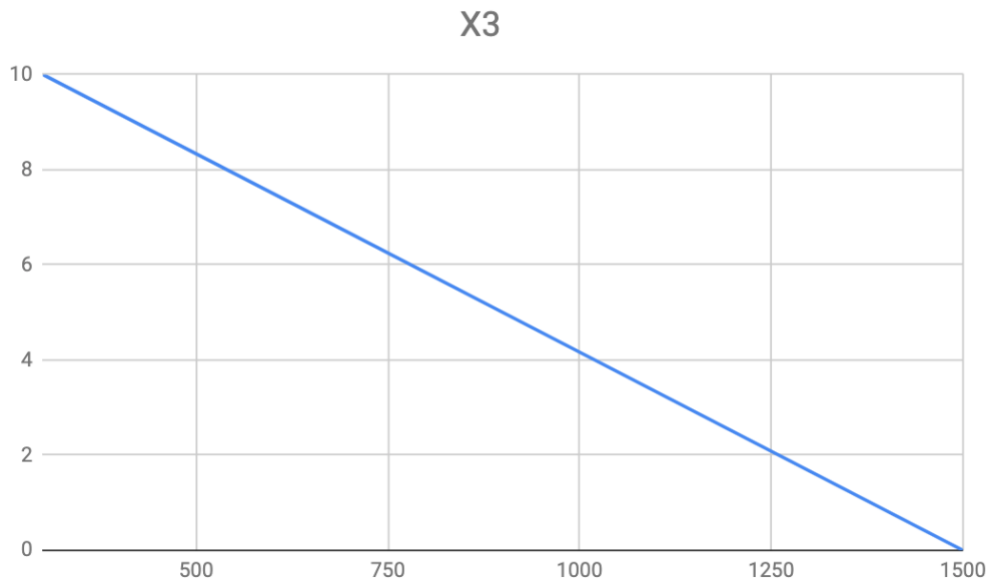


Рисунок 5.4 – X3, час обробки даних алгоритмом.

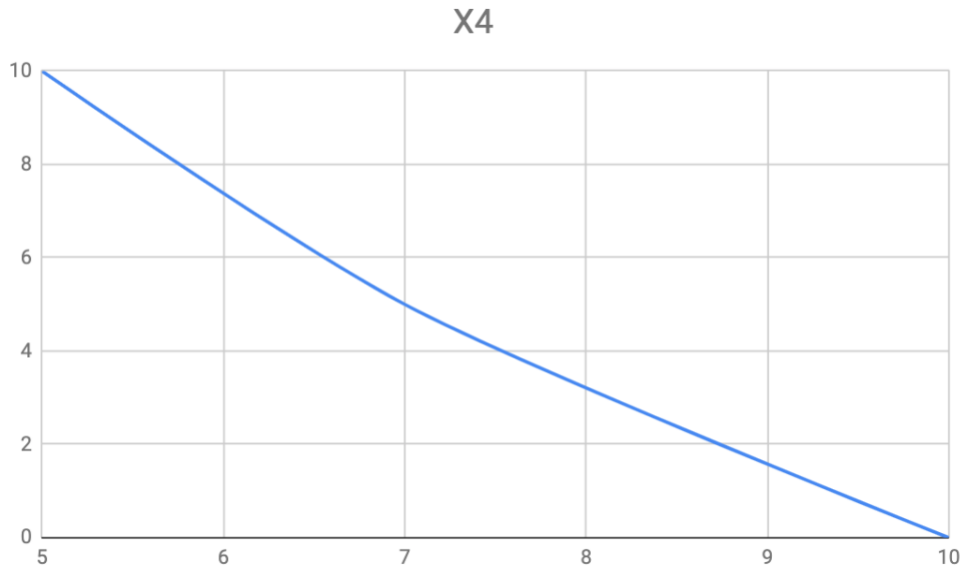


Рисунок 5.5 – X4, кількість додатково імпортованих бібліотек.

5.4. Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

1. визначення рівня значимості параметра шляхом присвоєння різних рангів;
2. перевірку придатності експертних оцінок для подальшого використання;
3. визначення оцінки попарного пріоритету параметрів;
4. обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 9.

Таблиця 9 – Результати ранжування параметрів

Параметр	Назва параметра	Вимір	Ранг параметра за оцінкою експерта							Сума рангів	Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Точність розв’язку	%	1	2	1	1	1	1	1	8	-9.5	90.25
X2	Потенційний об'єм програмного коду	кількість рядків коду	3	3	4	4	3	3	4	24	6.5	42.25
X3	Час обробки даних алгоритмом	с	2	1	2	3	2	2	2	14	-3.5	12.25
X4	Кількість додатково імпортованих бібліотек	одиниці	4	4	3	2	4	4	3	24	6.5	42.25
	Разом		10	10	10	10	10	10	10	70	0	187

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70,$$

де N – число експертів,

n – кількість параметрів.

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17.5.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T.$$

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^{\,2} = 187.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12*187}{49(64-4)} = \frac{2244}{2940} = 0.76 > W_k = 0.67.$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний. Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 10.

Таблиця 10 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	<	<	<	<	0.5
X1 і X3	<	>	<	<	<	<	<	<	0.5
X1 і X4	<	<	<	<	<	<	<	<	0.5
X2 і X3	>	>	>	>	>	>	>	>	1.5
X2 і X4	<	<	>	>	<	<	>	<	0.5
X3 і X4	<	<	<	>	<	<	<	<	0.5

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$. Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$K_{\text{ві}} = \frac{\sum_{i=1}^{N} a_{ij}}{\sum_{i=1}^{N}\sum_{i=1}^{N} a_{ij}} .$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{\sum_{i=1}^{N} a_{ij} b_j}{\sum_{i=1}^{N}\sum_{i=1}^{N} a_{ij} K_j} .$$

Як видно з таблиці 11, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 11 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітерація		Друга ітерація		Третя ітерація	
	1	2	3	4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1	0.5	0.5	0.5	2.5	0.156	9.25	0.156	34.175	0.158
X2	1.5	1	1.5	0.5	4.5	0.281	16.25	0.275	59.125	0.273

Продовження таблиці 11

X3	1.5	0.5	1	0.5	3.5	0.218	12.25	0.207	44.875	0.207
X4	1.5	1.5	1.5	1	5.5	0.345	21.25	0.36	77.875	0.36
Всього					16	1	59	1	216	1

5.5. Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій.

Таблиця 12 – Розрахунок показників рівня якості варіантів реалізації основних функцій

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	A	X4	6	9	0.36	3.24
F2	A	X1	4	8	0.158	1.264
		X3	900	5	0.207	1.035

Продовження таблиці 12

F3	А	Х2	700	6	0.273	1.638
	Б	Х2	1000	3	0.273	0.819

За даними з таблиці 12 визначаємо рівень якості кожного з варіантів:

$$К_{К1} = 3.24 + 1.264 + 1.035 + 1.638 = 7.177.$$
$$К_{К2} = 3.24 + 1.264 + 1.035 + 0.819 = 6.358.$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

25.6. Економічний аналіз варіантів розробки програмного продукту

Для визначення вартості розробки програмного продукту спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3. Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних. Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань. Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$$Т_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328{,}64 \text{ людино-годин.}$$
$$Т_{II} = (122.4 + 19.44 + 6.91 + 19.44) \cdot 8 = 1345.52 \text{ людино-годин.}$$

Найбільш високу трудомісткість має другий варіант.

В розробці беруть участь два програмісти з окладом 9000 грн., один фінансовий аналітик з окладом 12000грн. Визначимо середню зарплату за годину за формулою:

$$C_{\text{ч}} = \frac{M}{t \cdot T_m},$$

де М – місячний оклад працівників,

t – кількість робочих днів тиждень,

T_m – кількість робочих годин в день.

$$C_{\text{ч}} = \frac{9000+9000+12000}{3*21*8} = 59.52 \text{ грн.}$$

Розрахуємо заробітну плату.

$$C_{\text{ЗП}} = 59.52 \cdot 1328.64 \cdot 1.2 = 94896.78 \text{ грн.}$$
$$C_{\text{ЗП}} = 59.52 \cdot 1345.52 \cdot 1.2 = 96102.42 \text{ грн.}$$

Відрахування на соціальний внесок становить 22%:

$$C_{\text{ВІД}} = C_{\text{ЗП}} \cdot 0.22 = 82253.45 \cdot 0.22 = 20877.29 \text{ грн.}$$
$$C_{\text{ВІД}} = C_{\text{ЗП}} \cdot 0.22 = 83298.45 \cdot 0.22 = 21142.53 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. ($C_{\text{М}}$). Оскільки одна ЕОМ обслуговує одного програміста з окладом 9000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 9000 \cdot 0,2 = 21600 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 21600 \ (1 + 0.2) = 25920 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 17280 \cdot 0,22 = 5702,4 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$$С_А = 1.15 \cdot 0.25 \cdot 10000 = 2875 \text{ грн.}$$

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = 1.15 \cdot 10000 \cdot 0.05 = 575 \text{ грн.}$$

Ефективний годинний фонд часу за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 11 – 16) \cdot 8 \cdot 0.9 =$$
$$= 1684.8 \text{ годин,}$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1684.8 \cdot 0,3 \cdot 2,28974 \cdot 2.7515 = 3184.38 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 10000 \cdot 0,67 = 6700 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н = 25920 + 5702,4 + 2875 + 575 + 3184.38 + 6700 = 44956.78 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 44956.78/1684.8 = 26.69 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т,$$

$$С_М = 26.69 \cdot 1328,64 = 35461.40 \text{ грн.}$$

$$С_М = 26.69 \cdot 1345.52 = 35911.92 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0.67,$$

$$С_Н = 94896.78 \cdot 0,67 = 63580.84 \text{ грн.}$$

$$С_Н = 96102.42 \cdot 0.67 = 64388.62 \text{ грн.}$$

Отже, вартість розробки програмного продукту за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н,$$

$$С_{ПП} = 94896.78 + 20877.29 + 35461.40 + 63580.84 = 214816.31 \text{ грн.}$$

$$С_{ПП} = 96102.42 + 21142.53 + 35911.92 + 64388.62 = 217545.49 \text{ грн.}$$

5.7. Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$$К_{ТЕР1} = 7.177 / 214816.31 = 0,32 \cdot 10^{-4},$$

$$К_{ТЕР2} = 6.358 / 217532.04 = 0,29 \cdot 10^{-4}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0,32 \cdot 10^{-4}$. Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0,32 \cdot 10^{-4}$. Цей варіант реалізації програмного продукту має такі параметри: мова програмування – Python, бібліотека для глибинного навчання Keras та бібліотека візуалізації Matplotlib.

.

ВИСНОВКИ

Для інвесторів управління капіталом стає все більш важливим сьогодні. Професійні інвестиційні менеджери, як і індивідуальні інвестори, всі прагнуть мати ефективні інструменти для розуміння тенденцій фондового ринку, мінімізувати інвестиційний ризик і збільшити їх прибуток. Деякі вважають, що дуже складно передбачити ціни на акції. Однак, в реальному діловому світі, однак, трейдери успішно виконують тисячі транзакцій щороку, які навряд чи можна вважати суто спекулятивними.

З самого початку фінансових операцій на біржі люди розробляли методи для прогнозування вартості активів у майбутньому. З розвитком обчислювальних технологій та штучного інтелекту, точність методів зростає кожного дня.

У цьому дослідженні порівнювалися показники прогнозування між нейромережею та методом прогнозування класичних часових рядів, а саме вартістю акцій компанії на фондовій біржі. Моделі нейронних мереж, наведені в даному дослідженні, показали дуже набагато кращу здатність точного прогнозування, і отже є перспективним напрямком розвитку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Lawrence R. Using Neural Networks to Forecast Stock Market Prices. New York; London: Oxford University Press, 1997. 326 p.
2. Бідюк П.І. Аналіз часових рядів: навчальний посібник. К: Політехніка, 2010. 317 с.
3. Steel A. Predictions in Financial Time Series Data: Doctoral dissertation / University at Albany. Albany, NY, 2014. 192 p.
4. Adhikari R. An Introductory Study on Time Series Modeling and Forecasting. *Journal of Artificial Intelligence.* 2009. Vol. 42, No. 5. P.856–874.
5. Nwankpa C., Ijomah W. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. *Journal of Artificial Intelligence*. 2018. Вип. 1 (9). URL: https://arxiv.org/pdf/1811.03378.pdf (дата звернення: 17.04.2019).
6. Robert J. Van Eyden The Application of Neural Networks in the Forecasting of Share Prices. New York: Finance and Technology Publishing, 1996. 326 p.
7. Yoon Y., G. Swales Applying Artificial Neural Networks to Investment Analysis. London: Taylor & Francis, 2011. 80 с.
8. Jaybhay K. Stock Market Prediction Model by Combining Numeric and News Textual Mining. *International Journal of Computer Applications*. 2012. Vol. 6, No. 19. P. 18–20.
9. Володин С. Н. Прогнозирование динамики курсовых стоимостей акций фондового рынка с применением резонансных систем искусственного интеллекта: дис. ... канд. техн. наук: 01.05.03 / Высшая школа экономики. Москва, 2017. 113 с.

10. Di Persio L. Artificial Neural Networks architectures for stock price prediction: comparisons and applications. *INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING*. 2016. Vol. 31, No. 10. P. 404–405.
11. K. Kamijo, T. Tanigawa. Stock price pattern recognition: A recurrent neural network approach. In Neural Networks in Finance and Investing. New Delhi: Probus Publishing Company, 1993. 370 c.
12. Wolfgang G., Sascha S. Predicting Time Series with Space-Time Convolutional and Recurrent Neural Networks. *ESANN 2017*: Proceedings of the technical sessions presented at the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, 26-28 April, 2017. Munich: Schwabing, 2018. P. 71–85.
13. Крейдич. Н.В., Семенченко Н.В., Рощина Л.С., Борданова Н.Ю. Рекомендації до виконання економіко-організаційного розділу дипломних робіт для студентів всіх технічних спеціальностей. *Економіка та організація виробництва*: навч. посіб. для студ. всіх технічних спеціальностей. 2018. URL: http://ela.kpi.ua/jspui/handle/123456789/26580 (дата звернення: 19.04.2019).

ДОДАТОК А ВХІДНІ ДАНІ ВАРТОСТІ АКЦІЙ

Таблиця А – Дані з фондової біржі про вартість акцій компанії Google

date	close	volume	open	high	low
5/10/19	1164.27	1314546	1162.38	1172.6	1142.5
5/9/19	1162.38	1185973	1166.27	1169.66	1150.85
5/8/19	1166.27	1309514	1174.1	1180.424	1165.74
5/7/19	1174.1	1551368	1189.39	1190.44	1161.04
5/6/19	1189.39	1563943	1185.4	1190.85	1166.26
5/3/19	1185.4	1980653	1162.61	1186.8	1169
5/2/19	1162.61	1944817	1168.08	1174.19	1155.002
5/1/19	1168.08	2642983	1188.48	1188.05	1167.18
4/30/19	1188.48	6194691	1185	1192.81	1175
4/29/19	1287.58	2412788	1274	1289.27	1266.2949
4/26/19	1272.18	1228276	1269	1273.07	1260.32
4/25/19	1263.45	1099614	1264.77	1267.4083	1252.03
4/24/19	1256	1015006	1264.12	1268.01	1255
4/23/19	1264.55	1271195	1250.69	1269	1246.38
4/22/19	1248.84	806577	1235.99	1249.09	1228.31

4/18/19	1236.37	1315676	1239.18	1242	1234.61
4/17/19	1236.34	1211866	1233	1240.56	1227.82
4/16/19	1227.13	855258	1225	1230.82	1220.12
4/15/19	1221.1	1187353	1218	1224.2	1209.1101
4/12/19	1217.87	926799	1210	1218.35	1208.11
4/11/19	1204.62	709417	1203.96	1207.96	1200.13
4/10/19	1202.16	724524	1200.68	1203.785	1196.435
4/9/19	1197.25	865416	1196	1202.29	1193.08
4/8/19	1203.84	859969	1207.89	1208.69	1199.86
4/5/19	1207.15	900950	1214.99	1216.22	1205.03
4/4/19	1215	949962	1205.94	1215.67	1204.13
4/3/19	1205.92	1014195	1207.48	1216.3	1200.5
4/2/19	1200.49	800820	1195.32	1201.35	1185.71
4/1/19	1194.43	1188235	1184.1	1196.66	1182
3/29/19	1173.31	1269573	1174.9	1178.99	1162.88
3/28/19	1168.49	966843	1171.54	1171.565	1159.4312
3/27/19	1173.02	1362217	1185.5	1187.559	1159.37

3/26/19	1184.62	1894639	1198.53	1202.83	1176.72
3/25/19	1193	1493841	1196.93	1206.3975	1187.04
3/22/19	1205.5	1668910	1226.32	1230	1202.825
3/21/19	1231.54	1195899	1216	1231.79	1213.15
3/20/19	1223.97	2089367	1197.35	1227.14	1196.17
3/19/19	1198.85	1404863	1188.81	1200	1185.87
3/18/19	1184.26	1212506	1183.3	1190	1177.4211
3/15/19	1184.46	2457597	1193.38	1196.57	1182.61
3/14/19	1185.55	1150950	1194.51	1197.88	1184.48
3/13/19	1193.32	1434816	1200.645	1200.93	1191.94
3/12/19	1193.2	2012306	1178.26	1200	1178.26
3/11/19	1175.76	1569332	1144.45	1176.19	1144.45
3/8/19	1142.32	1212271	1126.73	1147.08	1123.3
3/7/19	1143.3	1166076	1155.72	1156.755	1134.91
3/6/19	1157.86	1094100	1162.49	1167.5658	1155.49
3/5/19	1162.03	1422357	1150.06	1169.61	1146.195
3/4/19	1147.8	1444774	1146.99	1158.2804	1130.69

3/1/19	1140.99	1447454	1124.9	1142.97	1124.75
2/28/19	1119.92	1541068	1111.3	1127.65	1111.01
2/27/19	1116.05	968362	1106.95	1117.98	1101
2/26/19	1115.13	1469761	1105.75	1119.51	1099.92
2/25/19	1109.4	1395281	1116	1118.54	1107.27
2/22/19	1110.37	1048361	1100.9	1111.24	1095.6
2/21/19	1096.97	1414744	1110.84	1111.94	1092.52
2/20/19	1113.8	1080144	1119.99	1123.41	1105.28
2/19/19	1118.56	1046315	1110	1121.89	1110
2/15/19	1113.65	1442461	1130.08	1131.67	1110.65
2/14/19	1121.67	941678	1118.05	1128.23	1110.445
2/13/19	1120.16	1048630	1124.99	1134.73	1118.5
2/12/19	1121.37	1608658	1106.8	1125.295	1105.85
2/11/19	1095.01	1063825	1096.95	1105.945	1092.86
2/8/19	1095.06	1072031	1087	1098.91	1086.55
2/7/19	1098.71	2040615	1104.16	1104.84	1086
2/6/19	1115.23	2101674	1139.57	1147	1112.77

2/5/19	1145.99	3529974	1124.84	1146.85	1117.248
2/4/19	1132.8	2518184	1112.66	1132.8	1109.02
2/1/19	1110.75	1455609	1112.4	1125	1104.89
1/31/19	1116.37	1531463	1103	1117.33	1095.41
1/30/19	1089.06	1241760	1068.43	1091	1066.85
1/29/19	1060.62	1006731	1072.68	1075.15	1055.8647
1/28/19	1070.08	1277745	1080.11	1083	1063.8
1/25/19	1090.99	1114785	1085	1094	1081.82
1/24/19	1073.9	1317718	1076.48	1079.475	1060.7
1/23/19	1075.57	956526	1077.35	1084.93	1059.75
1/22/19	1070.52	1607398	1088	1091.51	1063.47
1/18/19	1098.26	1933754	1100	1108.352	1090.9
1/17/19	1089.9	1223674	1079.47	1091.8	1073.5
1/16/19	1080.97	1320530	1080	1092.375	1079.34
1/15/19	1077.15	1452238	1050.17	1080.05	1047.34
1/14/19	1044.69	1127417	1046.92	1051.53	1041.255
1/11/19	1057.19	1512651	1063.18	1063.775	1048.48

1/10/19	1070.33	1444976	1067.66	1071.15	1057.71
1/9/19	1074.66	1198369	1081.65	1082.63	1066.4
1/8/19	1076.28	1748371	1076.11	1084.56	1060.53
1/7/19	1068.39	1978077	1071.5	1073.9999	1054.76
1/4/19	1070.71	2080144	1032.59	1070.84	1027.4179
1/3/19	1016.06	1829379	1041	1056.98	1014.07
1/2/19	1045.85	1516681	1016.57	1052.32	1015.71
12/31/18	1035.61	1492541	1050.96	1052.7	1023.59
12/28/18	1037.08	1399218	1049.62	1055.56	1033.1
12/27/18	1043.88	2102069	1017.15	1043.89	997
12/26/18	1039.46	2337212	989.01	1040	983
12/24/18	976.22	1590328	973.9	1003.54	970.11
12/21/18	979.54	4560424	1015.3	1024.02	973.69
12/20/18	1009.41	2659047	1018.13	1034.22	996.36
12/19/18	1023.01	2419322	1033.99	1062	1008.05
12/18/18	1028.71	2101854	1026.09	1049.48	1021.44
12/17/18	1016.53	2337631	1037.51	1053.15	1007.9

12/14/18	1042.1	1685802	1049.98	1062.6	1040.79
12/13/18	1061.9	1329198	1068.07	1079.7597	1053.93
12/12/18	1063.68	1523276	1068	1081.65	1062.79
12/11/18	1051.75	1354751	1056.49	1060.6	1039.84
12/10/18	1039.55	1793465	1035.05	1048.45	1023.29
12/7/18	1036.58	2098526	1060.01	1075.26	1028.5
12/6/18	1068.73	2758098	1034.26	1071.2	1030.7701
12/4/18	1050.82	2278200	1103.12	1104.42	1049.98
12/3/18	1106.43	1900355	1123.14	1124.65	1103.6645
11/30/18	1094.43	2554416	1089.07	1095.57	1077.88
11/29/18	1088.3	1403540	1076.08	1094.245	1076
11/28/18	1086.23	2399374	1048.76	1086.84	1035.76
11/27/18	1044.41	1801334	1041	1057.58	1038.49
11/26/18	1048.62	1846430	1038.35	1049.31	1033.91
11/23/18	1023.88	691462	1030	1037.59	1022.3992
11/21/18	1037.61	1531676	1036.76	1048.56	1033.47
11/20/18	1025.76	2447254	1000	1031.74	996.02

11/19/18	1020	1837207	1057.2	1060.79	1016.2601
11/16/18	1061.49	1641232	1059.41	1067	1048.98
11/15/18	1064.71	1819132	1044.71	1071.85	1031.78
11/14/18	1043.66	1561656	1050	1054.5643	1031
11/13/18	1036.05	1496534	1043.29	1056.605	1031.15
11/12/18	1038.63	1429319	1061.39	1062.12	1031
11/9/18	1066.15	1343154	1073.99	1075.56	1053.11
11/8/18	1082.4	1463022	1091.38	1093.27	1072.2048
11/7/18	1093.39	2057155	1069	1095.46	1065.9
11/6/18	1055.81	1225197	1039.48	1064.345	1038.07
11/5/18	1040.09	2436742	1055	1058.47	1021.24
11/2/18	1057.79	1829295	1073.73	1082.975	1054.61
11/1/18	1070	1456222	1075.8	1083.975	1062.46
10/31/18	1076.77	2528584	1059.81	1091.94	1057
10/30/18	1036.21	3209126	1008.46	1037.49	1000.75
10/29/18	1020.08	3873644	1082.47	1097.04	995.83
10/26/18	1071.47	4185201	1037.03	1106.53	1034.09

10/25/18	1095.57	2511884	1071.79	1110.98	1069.55
10/24/18	1050.71	1910060	1104.25	1106.12	1048.74
10/23/18	1103.69	1847798	1080.89	1107.89	1070
10/22/18	1101.16	1494285	1103.06	1112.23	1091
10/19/18	1096.46	1264605	1093.37	1110.36	1087.75
10/18/18	1087.97	2056606	1121.84	1121.84	1077.09
10/17/18	1115.69	1397613	1126.46	1128.99	1102.19
10/16/18	1121.28	1845491	1104.59	1124.22	1102.5
10/15/18	1092.25	1343231	1108.91	1113.4464	1089
10/12/18	1110.08	2029872	1108	1115	1086.402
10/11/18	1079.32	2939514	1072.94	1106.4	1068.27
10/10/18	1081.22	2574985	1131.08	1132.17	1081.13
10/9/18	1138.82	1308706	1146.15	1154.35	1137.572
10/8/18	1148.97	1877142	1150.11	1168	1127.3636
10/5/18	1157.35	1184245	1167.5	1173.4999	1145.12
10/4/18	1168.19	2151762	1195.33	1197.51	1155.576
10/3/18	1202.95	1207280	1205	1206.41	1193.83

10/2/18	1200.11	1655602	1190.96	1209.96	1186.63
10/1/18	1195.31	1345250	1199.89	1209.9	1190.3
9/28/18	1193.47	1306822	1191.87	1195.41	1184.5
9/27/18	1194.64	1244278	1186.73	1202.1	1183.63
9/26/18	1180.49	1346434	1185.15	1194.23	1174.765
9/25/18	1184.65	937577	1176.15	1186.88	1168
9/24/18	1173.37	1218532	1157.17	1178	1146.91
9/21/18	1166.09	4363929	1192	1192.21	1166.04
9/20/18	1186.87	1209855	1179.99	1189.89	1173.36
9/19/18	1171.09	1185321	1164.98	1173.21	1154.58
9/18/18	1161.22	1184407	1157.09	1176.08	1157.09
9/17/18	1156.05	1279147	1170.14	1177.24	1154.03
9/14/18	1172.53	934300	1179.1	1180.425	1168.3295
9/13/18	1175.33	1402005	1170.74	1178.61	1162.85
9/12/18	1162.82	1291304	1172.72	1178.61	1158.36
9/11/18	1177.36	1209171	1161.63	1178.68	1156.24
9/10/18	1164.64	1115259	1172.19	1174.54	1160.11

9/7/18	1164.83	1401034	1158.67	1175.26	1157.215
9/6/18	1171.44	1886690	1186.3	1186.3	1152
9/5/18	1186.48	2043732	1193.8	1199.0096	1162
9/4/18	1197	1800509	1204.27	1212.99	1192.5
8/31/18	1218.19	1812366	1234.98	1238.66	1211.2854
8/30/18	1239.12	1320261	1244.23	1253.635	1232.59
8/29/18	1249.3	1295939	1237.45	1250.66	1236.3588
8/28/18	1231.15	1296532	1241.29	1242.545	1228.69
8/27/18	1241.82	1154962	1227.6	1243.09	1225.716
8/24/18	1220.65	946529	1208.82	1221.65	1206.3588
8/23/18	1205.38	988509	1207.14	1221.28	1204.24
8/22/18	1207.33	881463	1200	1211.84	1199
8/21/18	1201.62	1187884	1208	1217.26	1200.3537
8/20/18	1207.77	864462	1205.02	1211	1194.6264
8/17/18	1200.96	1381724	1202.03	1209.02	1188.24
8/16/18	1206.49	1319985	1224.73	1225.9999	1202.55
8/15/18	1214.38	1815642	1229.26	1235.24	1209.51

8/14/18	1242.1	1342534	1235.19	1245.8695	1225.11
8/13/18	1235.01	957153	1236.98	1249.2728	1233.6405
8/10/18	1237.61	1107323	1243	1245.695	1232
8/9/18	1249.1	805227	1249.9	1255.542	1246.01
8/8/18	1245.61	1369650	1240.47	1256.5	1238.0083
8/7/18	1242.22	1493073	1237	1251.17	1236.17
8/6/18	1224.77	1080923	1225	1226.0876	1215.7965
8/3/18	1223.71	1072524	1229.62	1230	1215.06
8/2/18	1226.15	1520488	1205.9	1229.88	1204.79
8/1/18	1220.01	1567142	1228	1233.47	1210.21
7/31/18	1217.26	1632823	1220.01	1227.5877	1205.6
7/30/18	1219.74	1822782	1228.01	1234.916	1211.47
7/27/18	1238.5	2115802	1271	1273.89	1231
7/26/18	1268.33	2334881	1251	1269.7707	1249.02
7/25/18	1263.7	2115890	1239.13	1265.86	1239.13
7/24/18	1248.08	3303268	1262.59	1266	1235.56
7/23/18	1205.5	2584034	1181.01	1206.49	1181

7/20/18	1184.91	1246898	1186.96	1196.86	1184.22
7/19/18	1186.96	1256113	1191	1200	1183.32
7/18/18	1195.88	1391232	1196.56	1204.5	1190.34
7/17/18	1198.8	1585091	1172.22	1203.04	1170.6
7/16/18	1183.86	1049560	1189.39	1191	1179.28
7/13/18	1188.82	1221687	1185	1195.4173	1180
7/12/18	1183.48	1251083	1159.89	1184.41	1155.935
7/11/18	1153.9	1094301	1144.59	1164.29	1141.0003
7/10/18	1152.84	789249	1156.98	1159.59	1149.59
7/9/18	1154.05	906073	1148.48	1154.67	1143.42
7/6/18	1140.17	966155	1123.58	1140.93	1120.7371
7/5/18	1124.27	1060752	1110.53	1127.5	1108.48
7/3/18	1102.89	679034	1135.82	1135.82	1100.02
7/2/18	1127.46	1188616	1099	1128	1093.8
6/29/18	1115.65	1275979	1120	1128.2265	1115
6/28/18	1114.22	1072438	1102.09	1122.31	1096.01
6/27/18	1103.98	1287698	1121.34	1131.8362	1103.62

6/26/18	1118.46	1559791	1128	1133.21	1116.6589
6/25/18	1124.81	2155276	1143.6	1143.91	1112.78
6/22/18	1155.48	1310164	1159.14	1162.4965	1147.26
6/21/18	1157.66	1232352	1174.85	1177.295	1152.232
6/20/18	1169.84	1648248	1175.31	1186.2856	1169.16
6/19/18	1168.06	1616125	1158.5	1171.27	1154.01
6/18/18	1173.46	1400641	1143.65	1174.31	1143.59
6/15/18	1152.26	2119134	1148.86	1153.42	1143.485
6/14/18	1152.12	1350085	1143.85	1155.47	1140.64
6/13/18	1134.79	1490017	1141.12	1146.5	1133.38

ДОДАТОК Б ТЕКСТ ПРОГРАМИ

В.1 Програмний код реалізації методів згладжування

```
import numpy as np                    # vectors and matrices
import pandas as pd                   # tables and data manipulations
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error
%matplotlib inline
from statsmodels.tsa.holtwinters import ExponentialSmoothing, SimpleExpSmoothing,
Holt
import statsmodels.api as sm
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.ar_model import AR
#Subsetting the dataset
#Index 11856 marks the end of year 2013
stock_data = pd.read_csv('stock10y.csv')
import math
stock_data["average"] = (stock_data["high"] + stock_data["low"])/2
stock_data.head()
#Aggregating the dataset at daily level
# stock_data.Timestamp = pd.to_datetime(stock_data.date,format='%m/%d/%y')
stock_data.index = pd.to_datetime(stock_data.date,format='%m/%d/%y')
from sklearn import preprocessing


x = stock_data.average.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
```

```
x_scaled = min_max_scaler.fit_transform(x.reshape(-1, 1))
stock_data.average = x_scaled
#Creating train and test set
n_test = int(0.3*len(stock_data))
test=stock_data[0:n_test]
train =stock_data[n_test:]
plt.plot(train.average, label = 'Train')
plt.plot(test.average, label = 'Test')
plt.title("Share prices")
plt.legend()
plt.xlabel("Time")
plt.show()
window = [30,10,5]
plt.figure(figsize=(16,8))
plt.plot(train['average'], label='Train')
plt.plot(test['average'], label='Test')
for w in window:
    y_hat = ((stock_data['average'].iloc[::-1]).rolling(w).mean()).iloc[::-1]
    plt.plot(y_hat[0:n_test], label='Moving average forecast window='+str(w))
    mse = mean_squared_error(test.average, y_hat[0:n_test])
    mae = mean_absolute_error(test.average, y_hat[0:n_test])
    residual = test.average - y_hat[0:n_test]
    print('Moving Average: window =',str(w),'\nMSE = ',mse,'\nMAE = ',mae,'\n')
plt.legend(loc='best')
plt.show()
```

```
print ('**********************\n RESIDUALS FOR TEST')
plt.figure(figsize=(16,8))
plt.plot(residual)
parameters = [12,1,0.5]
plt.figure(figsize=(16,8))
plt.plot(train['average'], label='Train')
plt.plot(test['average'], label='Test')
for p in parameters:
    y_hat = pd.ewma(stock_data.average, halflife=p)
#     print(y_hat)
    plt.plot(y_hat[0:n_test], label='Moving average forecast halflife='+str(p))
    mse = mean_squared_error(test.average, y_hat[0:n_test])
    mae = mean_absolute_error(test.average, y_hat[0:n_test])
    residual = test.average - y_hat[0:n_test]
    print('Weighted Average: halflife =',str(p),'\nMSE = ',mse,'\nMAE = ',mae,'\n')
plt.legend(loc='best')
plt.show()
print ('**********************\n RESIDUALS FOR TEST')
plt.figure(figsize=(16,8))
plt.plot(residual)
parameters = [0.1,0.2,0.6]
plt.figure(figsize=(16,8))
plt.plot(train['average'], label='Train')
plt.plot(test['average'], label='Test')
for p in parameters:
```

```
                                                                fit1 =
SimpleExpSmoothing(stock_data.average).fit(smoothing_level=p,optimized=False)
    y_hat = fit1.fittedvalues
    plt.plot(y_hat[0:n_test], label='Simple exp smoothing level='+str(p))
    mse = mean_squared_error(test.average, y_hat[0:n_test])
    mae = mean_absolute_error(test.average, y_hat[0:n_test])
    residual = test.average - y_hat[0:n_test]
    print('Simple exp smoothing: level =',str(p),'\nMSE = ',mse,'\nMAE = ',mae,'\n')
plt.legend(loc='best')
plt.show()
print ('**********************\n RESIDUALS FOR TEST')
plt.figure(figsize=(16,8))
plt.plot(residual)
import statsmodels.api as sm

from statsmodels.tsa.seasonal import seasonal_decompose

result = seasonal_decompose(stock_data.average, model='additive',freq=30)
result.plot()
plt.show()

parameters = [[0.1,0.3],[0.2,0.8],[0.6,0.6]]
plt.figure(figsize=(16,8))
plt.plot(train['average'], label='Train')
plt.plot(test['average'], label='Test')
```

```
for p,s in parameters:
    fit1 = Holt(stock_data.average).fit(smoothing_level=p,smoothing_slope = s)
    y_hat = fit1.fittedvalues
    plt.plot(y_hat[0:n_test], label='Double exp smoothing level='+str(p)+' slope='+str(s))
    mse = mean_squared_error(test.average, y_hat[0:n_test])
    mae = mean_absolute_error(test.average, y_hat[0:n_test])
    residual = test.average - y_hat[0:n_test]
     print('Double exp smoothing: level =',str(p),'slope =',str(s),'\nMSE = ',mse,'\nMAE =
',mae,'\n')
plt.legend(loc='best')
plt.show()
print ('**********************\n RESIDUALS FOR TEST')
plt.figure(figsize=(16,8))
plt.plot(residual)
# parameters = [[6,'add','add'],[6,'add','mul'],[10,'add','add'],[10,'add','mul']]
parameters = [[10,'add','add'],[6,'add','add']]
plt.figure(figsize=(16,8))
plt.plot(train['average'], label='Train')
plt.plot(test['average'], label='Test')
for p,tr,seas in parameters:
        fit1 = ExponentialSmoothing(stock_data.average,seasonal_periods=p ,trend=tr,
seasonal=seas).fit()
    y_hat = fit1.fittedvalues
        plt.plot(y_hat[0:n_test], label='Holt-Winters periods='+str(p)+' trend='+str(tr)+'
seasonal='+str(seas))
```

```
    mse = mean_squared_error(test.average, y_hat[0:n_test])
    mae = mean_absolute_error(test.average, y_hat[0:n_test])
    residual = test.average - y_hat[0:n_test]
        print('Holt-Winters:  periods  =',str(p),'trend  =',str(tr),'seas  =',str(seas),'\nMSE  =
',mse,'\nMAE = ',mae,'\n')
plt.legend(loc='best')
plt.show()
print ('**********************\n RESIDUALS FOR TEST')
plt.figure(figsize=(16,8))
plt.plot(residual)
```

В.2 Програмний код реалізації методів авторегресії

```
import numpy as np                          # vectors and matrices
import pandas as pd                         # tables and data manipulations
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error
import warnings
import itertools
%matplotlib inline
import statsmodels.api as sm
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.ar_model import AR
#Subsetting the dataset
#Index 11856 marks the end of year 2013
```

```
stock_data = pd.read_csv('stock10y.csv')
import math
stock_data["average"] = (stock_data["high"] + stock_data["low"])/2
stock_data.head()
#Aggregating the dataset at daily level
# stock_data.Timestamp = pd.to_datetime(stock_data.date,format='%m/%d/%y')
stock_data.index = pd.to_datetime(stock_data.date,format='%m/%d/%y')
from sklearn import preprocessing

x = stock_data.average.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x.reshape(-1, 1))
stock_data.average = x_scaled
#Creating train and test set
n_test = int(0.3*len(stock_data))
test=stock_data[0:n_test]
train =stock_data[n_test:]
plt.plot(train.average, label = 'Train')
plt.plot(test.average, label = 'Test')
plt.title("Share prices")
plt.legend()
plt.xlabel("Time")
plt.show()
y = stock_data.average[::-1]
from pmdarima.arima import auto_arima
```

```
for el in range(1,3):
    stepwise_model = auto_arima(y, start_p=3, start_q=1,
                                max_p=4, max_q=5, m=12,
                                start_P=0, seasonal=False,
                                d=el, D=0, trace=True,
                                error_action='ignore',
                                suppress_warnings=True,
                                stepwise=True)
    print(stepwise_model.aic())

mod = sm.tsa.statespace.SARIMAX(y,
                                order=(2,0,0),
                                seasonal_order=(0,0,0,12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results_ar = mod.fit()
pred_ar = results_ar.get_prediction(start=len(train), dynamic=False)
pred_ci = pred_ar.conf_int()
print(results.summary().tables[1])
plt.figure(figsize=(16,8))
ax = train.average.plot(label='train',color='green')
ax = test.average.plot(label='test',color='blue')
pred_ar.predicted_mean.plot(ax=ax, label='AR(3)', alpha=.7, color = 'red')
```

```
# ax.fill_between(pred_ci.index,
#                 pred_ci.iloc[:, 0],
#                 pred_ci.iloc[:, 1], color='k', alpha=.2)

ax.set_xlabel('Time check')
ax.set_ylabel('Share price')
plt.legend()

plt.show()
y_forecasted = pred_ar.predicted_mean
plt.figure(figsize=(16,8))
# Compute the mean square error
mse = ((y_forecasted - test.average) ** 2).mean()
mae = (abs(y_forecasted - test.average)).mean()
print('TEST MSE {}'.format(round(mse, 6)))
print('TEST MAE {}'.format(round(mae, 6)))
residual_ar = y_forecasted-test.average
plt.plot(residual_ar[:-10])

mod = sm.tsa.statespace.SARIMAX(y,
                                order=(3,0,1),
                                seasonal_order=(0,0,0,12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
```

```
results_arma = mod.fit()

print(results.summary().tables[1])
pred_arma = results_arma.get_prediction(start=len(train), dynamic=False)
pred_ci = pred_arma.conf_int()
plt.figure(figsize=(16,8))
ax = train.average.plot(label='train',color='green')
ax = test.average.plot(label='test',color='blue')
pred_arma.predicted_mean.plot(ax=ax, label='ARMA(3,1)', alpha=.7, color = 'red')

# ax.fill_between(pred_ci.index,
#                 pred_ci.iloc[:, 0],
#                 pred_ci.iloc[:, 1], color='k', alpha=.2)

ax.set_xlabel('Time check')
ax.set_ylabel('Share price')
plt.legend()

plt.show()
y_forecasted = pred_arma.predicted_mean
plt.figure(figsize=(16,8))
# Compute the mean square error
mse = ((y_forecasted - test.average) ** 2).mean()
mae = (abs(y_forecasted - test.average)).mean()
print('TEST MSE {}'.format(round(mse, 6)))
```

```
print('TEST MAE {}'.format(round(mae, 6)))
residual_arma = y_forecasted-test.average
plt.plot(residual_arma[:-10])
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(2,1,1),
                                seasonal_order=(0,0,0,12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results_arima = mod.fit()

print(results.summary().tables[1])
pred_arima = results_arima.get_prediction(start=len(train), dynamic=False)
pred_ci = pred_arima.conf_int()
plt.figure(figsize=(16,8))
ax = train.average.plot(label='train',color='green')
ax = test.average.plot(label='test',color='blue')
pred_arima.predicted_mean.plot(ax=ax, label='ARIMA(2,1,1)', alpha=.7, color = 'red')

# ax.fill_between(pred_ci.index,
#                 pred_ci.iloc[:, 0],
#                 pred_ci.iloc[:, 1], color='k', alpha=.2)

ax.set_xlabel('Time check')
ax.set_ylabel('Share price')
```

```
plt.legend()


plt.show()
y_forecasted = pred_arima.predicted_mean
plt.figure(figsize=(16,8))
# Compute the mean square error
mse = ((y_forecasted - test.average) ** 2).mean()
mae = (abs(y_forecasted - test.average)).mean()
print('TEST MSE {}'.format(round(mse, 6)))
print('TEST MAE {}'.format(round(mae, 6)))
residual_arima = y_forecasted-test.average
plt.plot(residual_arima[:-10])
```

В.3 Програмний код реалізації багатошарового перцептрону Румельхарта

```
from math import sqrt
import numpy as np
import pandas as pd
from pandas import concat
from pandas import read_csv
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error,r2_score
from keras.models import Sequential
from keras.layers import Dense, Flatten, LSTM, ConvLSTM2D
from keras.layers.convolutional import Conv1D,MaxPooling1D
```

```
stock_data = pd.read_csv("stock10y.csv")
from numpy import array
from numpy import mean
from numpy import std

stock_data["average"] = (stock_data["high"] + stock_data["low"])/2
stock_data.index = pd.to_datetime(stock_data.date,format='%m/%d/%y')
stock_data.head()

m = stock_data.average.mean()
s = stock_data.average.std()
price = []
for x in stock_data.average:
    price.append((x - m)/s)
stock_data.average=price


test_size=int(.3 * len(stock_data))
plt.plot(stock_data.average[:test_size], color='orange',label = 'Train')
plt.plot(stock_data.average[test_size:], color='blue', label = 'Test')
plt.title("Stock Average Prices")
plt.xlabel("Time")
plt.ylabel("Stock Opening Price")
plt.legend()
plt.show()
```

```
# data split
n_test = int(0.3*len(stock_data))

# evaluate mlp
from math import sqrt
from numpy import array
from numpy import mean
from numpy import std
from pandas import DataFrame
from pandas import concat
from pandas import read_csv
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from matplotlib import pyplot

# split a univariate dataset into train/test sets
def train_test_split(data, n_test):
	return data[:-n_test], data[-n_test:]

# transform list into supervised learning format
def series_to_supervised(data, n_in=1, n_out=1):
	df = DataFrame(data)
	cols = list()
	# input sequence (t-n, ... t-1)
```

```
        for i in range(n_in, 0, -1):
                cols.append(df.shift(i))
        # forecast sequence (t, t+1, ... t+n)
        for i in range(0, n_out):
                cols.append(df.shift(-i))
        # put it all together
        agg = concat(cols, axis=1)
        # drop rows with NaN values
        agg.dropna(inplace=True)
        return agg.values

# root mean squared error or rmse
def measure_rmse(actual, predicted):
        return sqrt(mean_squared_error(actual, predicted))

# fit a model
def model_fit(train, config,opt,act):
        # unpack config
        n_input, n_nodes, n_epochs, n_batch = config
        # prepare data
        data = series_to_supervised(train, n_in=n_input)
        train_x, train_y = data[:, :-1], data[:, -1]
        # define model
        model = Sequential()
        model.add(Dense(n_nodes, activation=act, input_dim=n_input))
```

```
        model.add(Dense(1))
        model.compile(loss='mse', optimizer=opt, metrics=['mean_squared_error',
'mean_absolute_error'])
        # fit
        model.fit(train_x, train_y, epochs=n_epochs, batch_size=n_batch, verbose=2)
        return model

# forecast with a pre-fit model
def model_predict(model, history, config):
        # unpack config
        n_input, _, _, _ = config
        # prepare data
        x_input = array(history[-n_input:]).reshape(1, n_input)
        # forecast
        yhat = model.predict(x_input, verbose=0)
        return yhat[0]

# walk-forward validation for univariate data
def walk_forward_validation(data, n_test, cfg,opt,act):
    predictions = list()
    # split dataset
    train, test = train_test_split(data, n_test)
    # fit model
    model = model_fit(train, cfg,opt,act)
    # seed history with training dataset
```

```
    history = [x for x in train]
    # step over each time-step in the test set
    for i in range(len(test)):
        # fit model and make forecast for history
        yhat = model_predict(model, history, cfg)
        # store forecast in list of predictions
        predictions.append(yhat)
        # add actual observation to history for the next loop
        history.append(test[i])
    # estimate prediction error
    error_mse = mean_squared_error(test, predictions)
    error_mae = mean_absolute_error(test, predictions)
    #    print(' > %.3f' % error)
    return error_mse,error_mae


# repeat evaluation of a config
def repeat_evaluate(data, config, n_test,opt,act,n_repeats=5):
        # fit and evaluate the model n times
    scores_mse = []
    scores_mae = []
    for x in range(n_repeats):
        a,b = walk_forward_validation(data, n_test, config,opt,act)
        scores_mse.append(a)
        scores_mae.append(b)
    return scores_mse,scores_mae
```

```
# summarize model performance
def summarize_scores(name, scores_mse,scores_mae):
        # print a summary
    scores_m, score_std = mean(scores_mse), std(scores_mse)
    print('%s: %.3f MSE (+/- %.5f)' % (name, scores_m, score_std))
    scores_m, score_std = mean(scores_mae), std(scores_mae)
    print('%s: %.3f MAE (+/- %.5f)' % (name, scores_m, score_std))

# define config
config = [20, 10, 100, 100]
# grid search
scores_mse,scores_mae = repeat_evaluate(stock_data.average, config,
n_test,'adam','relu')
# summarize scores
summarize_scores('mlp', scores_mse,scores_mae)

# define config
config = [20, 20, 100, 100]
# grid search
scores_mse,scores_mae = repeat_evaluate(stock_data.average, config,
n_test,'adam','relu')
# summarize scores
summarize_scores('mlp', scores_mse,scores_mae)
```

```
# define config
config = [10, 20, 100, 100]
# grid search
scores_mse,scores_mae = repeat_evaluate(stock_data.average, config,
n_test,'adam','relu')
# summarize scores
summarize_scores('mlp', scores_mse,scores_mae)
```

В.4 Програмний код реалізації згорткових нейронних мереж

```
# evaluate lstm
from math import sqrt
import numpy as np
import pandas as pd
from pandas import concat
from pandas import read_csv
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error,r2_score
from keras.models import Sequential
from keras.layers import Dense, Flatten, LSTM, ConvLSTM2D
from keras.layers.convolutional import Conv1D,MaxPooling1D
stock_data = pd.read_csv("stock10y.csv")
from numpy import array
from numpy import mean
from numpy import std
```

```
stock_data["average"] = (stock_data["high"] + stock_data["low"])/2
stock_data.index = pd.to_datetime(stock_data.date,format='%m/%d/%y')
stock_data.head()

m = stock_data.average.mean()
s = stock_data.average.std()
price = []
for x in stock_data.average:
    price.append((x - m)/s)
stock_data.average=price

test_size=int(.3 * len(stock_data))
plt.plot(stock_data.average[:test_size], color='orange',label = 'Train')
plt.plot(stock_data.average[test_size:], color='blue', label = 'Test')
plt.title("Stock Average Prices")
plt.xlabel("Time")
plt.ylabel("Stock Opening Price")
plt.legend()
plt.show()

# data split
n_test = int(0.3*len(stock_data))

# split a univariate dataset into train/test sets
```

```
def train_test_split(data, n_test):
        return data[:-n_test], data[-n_test:]

# transform list into supervised learning format
def series_to_supervised(data, n_in=20, n_out=1):
    df = pd.DataFrame(data)
    cols = list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
    # put it all together
    agg = concat(cols, axis=1)
    # drop rows with NaN values
    agg.dropna(inplace=True)
    return agg.values

# fit a model
def model_fit(train, config,opt,act):
    n_input, n_filters, n_kernel, n_epochs, n_batch = config
    # prepare data
    data = series_to_supervised(train, n_in=n_input)
    train_x, train_y = data[:, :-1], data[:, -1]
```

```
    train_x = train_x.reshape((train_x.shape[0], train_x.shape[1], 1))
    model = Sequential()
            model.add(Conv1D(filters=n_filters, kernel_size=n_kernel, activation=act,
input_shape=(n_input, 1)))
    model.add(Conv1D(filters=n_filters, kernel_size=n_kernel, activation=act))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(1))
             model.compile(loss='mse', optimizer=opt, metrics=['mean_squared_error',
'mean_absolute_error'])
    model.summary()
    # fit
    h = model.fit(train_x, train_y, epochs=n_epochs, batch_size=n_batch, verbose=2)
    return model,h

# forecast with a pre-fit model
def model_predict(model, history, config):
    # unpack config
    n_input, _, _, _, _ = config
    # prepare data
    x_input = array(history[-n_input:]).reshape((1, n_input, 1))
    # forecast
    predicted_value = model.predict(x_input, verbose=2)
    return predicted_value[0]
```

```
# walk-forward validation for univariate data
def walk_forward_validation(data, n_train, cfg,opt,act):
    predictions = list()
    train, test = train_test_split(data, n_train)

    # fit model
    model,h = model_fit(train, cfg,opt,act)
    plt.plot(h.history['mean_squared_error'], label  ='mse')
    plt.plot(h.history['mean_absolute_error'], label  ='mae')
    plt.legend()
    plt.show()
    # seed history with training dataset
    history = [x for x in train]
    # plot metrics
    # step over each time-step in the test set
    for i in range(len(test)):
        # fit model and make forecast for history
        yhat = model_predict(model, history, cfg)
        # store forecast in list of predictions
        predictions.append(yhat)
        # add actual observation to history for the next loop
        history.append(test[i])

    print('TEST MSE:',mean_squared_error(test, predictions))
    print('TEST MAE:',mean_absolute_error(test, predictions))
```

```
    predictions = np.concatenate(predictions,axis=0)
#     plt.plot(predictions, color= 'red',label = 'train')
#     plt.plot(test, color= 'yellow',label = 'predicted')
#     plt.show()
    plt.plot(abs(test-predictions), label = 'Residual for test')
    plt.legend()
    return predictions,train


#         # step over each time-step in the test set
#         for i in range(len(test)):
#                 # fit model and make forecast for history
#                 yhat = model_predict(model, history, cfg)
#                 # store forecast in list of predictions
#                 predictions.append(yhat)
#                 # add actual observation to history for the next loop
#                 history.append(test[i])
#         # estimate prediction error
#         error = measure_rmse(test, predictions)
#         print(' > %.3f' % error)
#         return error

# n_train = int(0.7*len(stock_data))
config = [20, 300, 3, 200, 32]
result,tr = walk_forward_validation(stock_data.average[::-1], n_test,
config,'rmsprop','tanh')
```

```
stock_data['cnn'] = tr
stock_data.loc[:test_size,'cnn'] = result[::-1]
plt.plot(stock_data.average, color= 'green',label = 'test')
plt.plot(stock_data.average[:test_size:-1], color= 'red',label = 'train')
plt.plot(stock_data.cnn[test_size::-1], color= 'yellow',label = 'predicted')
plt.title("Price of stocks sold")
plt.xlabel("Time check")
plt.ylabel("Stock Price")
plt.legend()
plt.show()
```

В.5 Програмний код реалізації рекурентних нейронних мереж

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from keras import Sequential, backend
from keras.layers import Dense, LSTM, Activation
stock_data = pd.read_csv("stock10y.csv")
from sklearn.metrics import mean_squared_error, mean_absolute_error

plt.plot(stock_data.open[::-1], color='green', label = 'Open Price')
plt.title("Stock Open Prices")
```

```
plt.xlabel("Time")
plt.ylabel("USD")
plt.legend()
plt.show()

import math
stock_data["average"] = (stock_data["high"] + stock_data["low"])/2
stock_data.index = pd.to_datetime(stock_data.date,format='%m/%d/%y')
stock_data.head()

stock_data.describe()
input_feature= stock_data.iloc[:,[2,6]].values
input_data=input_feature

from sklearn.preprocessing import MinMaxScaler
sc= MinMaxScaler(feature_range=(0,1))
input_data[:,0:2] = sc.fit_transform(input_feature[:,:])

lookback= 10

test_size=int(.3 * len(stock_data))
X=[]
y=[]
for i in range(len(stock_data)-lookback-1):
    t=[]
```

```
    for j in range(0,lookback):
        t.append(input_data[[(i+j)], :])
    X.append(t)
    y.append(input_data[i+ lookback,1])
X, y= np.array(X), np.array(y)

X_test = X[:test_size]
y_test = y[:test_size]
X_train = X[test_size:]
y_train = y[test_size:]
X_train.shape,X_test.shape

X = X.reshape(X.shape[0],lookback, 2)
X_test = X_test.reshape(X_test.shape[0],lookback, 2)
X_train = X_train.reshape(X_train.shape[0],lookback, 2)

print(X.shape)
print(X_test.shape)
print(X_train.shape)

plt.plot(stock_data.average[:test_size], color='orange',label = 'Train')
plt.plot(stock_data.average[test_size:], color='blue', label = 'Test')
plt.title("Stock Average Prices")
plt.xlabel("Time")
plt.ylabel("Stock Opening Price")
```

```
plt.legend()
plt.show()

model = Sequential()
model.add(LSTM(units=30, return_sequences= True, input_shape=(X.shape[1],2)))
model.add(LSTM(units=30, return_sequences=True))
model.add(LSTM(units=30))
model.add(Dense(units=1))
model.summary()

model.compile(optimizer='sgd',
loss='mean_squared_error',metrics=['mean_squared_error', 'mean_absolute_error'])
history=model.fit(X_train, y_train, epochs=100, batch_size=32)

predicted_value_test= model.predict(X_test)
predicted_value_train= model.predict(X_train)

score = model.evaluate(X_test, y_test, verbose=0)
print('TEST LOSS:', score[0])
print('TEST MSE:',mean_squared_error(y_test, predicted_value_test))
print('TEST MAE:',mean_absolute_error(y_test, predicted_value_test))
plt.plot(input_data[:,1][::-1], color='green', label = 'dataset')
plt.plot(np.concatenate((predicted_value_test,  predicted_value_train))[::-1],  color=
'yellow',label = 'predicted')
```

```
plt.plot(np.concatenate((predicted_value_test,       predicted_value_train))[:test_size:-1],
color= 'red',label = 'train')
plt.title("Price of stocks sold")
plt.xlabel("Time check")
plt.ylabel("Stock Price")
plt.legend()
plt.show()

flat_list = [item for sublist in predicted_value_test for item in sublist]
```

ДОДАТОК В СТАТТІ

Кухарєв С. О., Олексієнко Г. О. Моделі прогнозування часових рядів на прикладі вартості акцій // Міжнародний науковий журнал "Інтернаука". — 2019. — №9.

Анотація. *В даній роботі розглянуті методи для прогнозування часових рядів – методи згладжування, авторегресії та нейронних мереж.*

Ключові слова: *часовий ряд, авторегресія, згладжування, нейронні мережі, згорткові нейронні мережі, рекурентні нейронні мережі.*

Аннотация. *В данной работе рассмотрены методы для прогнозирования временных рядов - методы сглаживания, авторегрессии и нейронных сетей.*

Ключевые слова: *временной ряд, авторегрессия, сглаживание, нейронные сети, сверточные нейронные сети, рекуррентные нейронные сети.*

Summary. *This paper describes methods for forecasting time series - smoothing methods, auto regression and neural networks.*

Key words: *time series, autoregression, smoothing, neural networks, convolutional neural networks, recurrent neural networks.*

Вступ. Фондовий ринок є однією з найважливіших сфер ринкової економіки, оскільки він надає компаніям доступ до капіталу, дозволяючи інвесторам купувати акції в компанії. Оскільки ціна акцій постійно коливається, передбачення того, як буде поводитись фінансовий ринок, є нагальною задачею для економістів. Задача є актуальною для всієї міжнародної економіки, оскільки можливість точного передбачення вартості акцій тісно пов’язано з отриманням

фінансового прибутку, а також зі зменшенням інвестиційного ризику та захисту інвестиційних прибутків від волатильності ринку.

Метою даної роботи є аналіз методів прогнозування часових рядів та виявлення на цій основі параметрів впливу на точність деяких моделей, що використовуються для прогнозування вартості акцій.

Для виконання роботи було взято дані про акції компанії Google з офіційного сайту американського біржового ринку NASDAQ за 5 років, починаючи з 1 квітня 2014 року. Датасет представляє з себе щоденну інформацію про ціну на акцій на початку та в кінці дня, максимальну та мінімальну вартість акції за день, та кількість проданих акцій.

В якості метрики для оцінки точності прогнозу для даної роботи було обрано метрики MAE (Mean Absolute Error) та MSE (Mean Squared Error), які можуть набувати значення від 0 до ∞, і не враховують напрямок помилок. Чим менше значення приймає показник, тим точнішим є прогноз. MAE вимірює середню абсолютну величину помилок у наборі прогнозів для безперервних змінних. MAE є лінійною оцінкою, що означає, що всі індивідуальні відмінності зважуються однаково в середньому. MSE є серенім квадратчним відхиленням похибки прогнозу. Оскільки помилки підносяться до квадрату перед тим, як вони усереднюються, MSE надає відносно високу вагу великим похибкам. Це означає, що MSE є найбільш корисним, коли великі помилки особливо небажані, що відповідає цілям даної роботи.

Згладжування – це важливий і широко поширений метод прогнозування фінансових ринків. Як правило, різні методи згладжування базуються на концепції ковзних середніх. Це допомагає зменшити вплив випадкового компонента у часовому ряді. Загальна формула для зваженого середнього

$$\hat{y_t} = \frac{1}{N} \sum_{i=1}^{N} w_i y_{k-i},$$

де N - число попередніх моментів часу, що було взяти до уваги при побудові прогнозу,

y_{k-i} - реальні значення показника в момент часу t_{k-i},

w_i - ваговий коефіцієнт для i-того компоненту ряду[1].

У випадку простого ковзного середнього вагові коефіцієнти дорівнюють одиниці, а у випадку експоненціально зваженого ковзного середнього коефіцієнти задаються як $\alpha(1-\alpha)^i, 0 < \alpha < 1$. Таким чином більший акцент робиться на останні точки даних.

Метод ковзного середнього було реалізовано для різних значень параметру N - кількості попередніх моментів часу, що було взяти до уваги при побудові прогнозу (рис. 1). Було перевірено, що при зменшенні довжини вікна N модель показує більш точний результат на тестовій вибірці, що вказує на те, що найостанніші є дані найбільш впливові при прогнозуванні, тобто прогноз при врахуванні останніх 5 днів є більш точним, ніж прогноз, в якому враховуються останні 20 днів.

Рисунок 1. Графік прогнозу простого ковзного середнього

Для експоненційного ковзного середнього (рис. 2) параметром є рівень згладжування - коефіцієнт α, що являє собою ступінь зменшення зважування від 0 до 1. Чим менший рівень згладжування, тим точніший є прогноз, оскільки кожне попереднє значення важить більше.

Рисунок 2. Графік прогнозу методом експоненціального ковзного середнього

Для даних з чітко вираженим трендом, що відповідає вхідним даним роботи, метод подвійного експоненціальне згладжування, що є рекурсивним застосуванням експоненційного фільтра двічі, дає кращий результат. Цей метод передбачав задання додатково параметру β, який відповідає за згладжування тренду (рис. 3). Комбінація пари α та β коригувало точність та якість прогнозу.

Рисунок 3. Графік прогнозу методом подвійного експоненціального ковзного середнього

Результати реалізації усіх методів згладжування зведені до таблиці 1.

Таблиця 1

Результати реалізації методів згладжування

Метод	Характерні параметри	Test MSE	Test MAE
Ковзне середнє	Вікно $N = 30$	0.0029	0.0455
	Вікно $N = 10$	0.0012	0.0281
	Вікно $N = 5$	0.0005	0.0180
Експоненціальне ковзне середнє	Рівень згладжування $\alpha = 0.1$	0.0018	0.0340
	Рівень згладжування $\alpha = 0.2$	0.0011	0.0259
	Рівень згладжування $\alpha = 0.6$	0.00049	0.0165
Подвійне експоненціальне ковзне середнє	$\alpha = 0.1,\ \beta = 0.3$	0.00244	0.0388
	$\alpha = 0.2,\ \beta = 0.8$	0.00163	0.0322
	$\alpha = 0.6,\ \beta = 0.6$	0.00054	0.0177

Модель авторегресії є ефективним інструментом для розуміння і прогнозування майбутніх значень часового ряду, яка включає в себе регресування змінної по значеннях ряду у минулому. Авторегресивні частини цих моделей описують, як послідовні спостереження в часі впливають один на одного, тоді як частини ковзних середніх захоплюють деякі можливі неспостережувані потрясіння.

Модель ARMA характеризує стохастичний процес за допомогою двох компонентів - авторегресії (AR) та ковзного середнього (MA). Частина AR передбачає регресування змінної на власні минулі значення. Частина MA включає моделювання похибки як лінійної комбінації похибок, що відбуваються в минулому. Позначення *ARMA(p, q)* характеризує модель з p авторегресійними компонентами і q компонентами для ковзного середнього:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + \sum_{i=1}^{q} b_i e_{n-i} + e_n ,$$

де a_i, b_i - параметри моделі,

e_n - білий шум.

Для застосування моделі ARIMA використовуються формули моделі ARMA, проте на вхід замість y_t подається $\Delta y_t = y_t - y_{y-1}$, і задається додатковий параметр, який вказує на кількість разів, коли до вхідних спостережень було застосовано диференціювання. Для розглянутих методів авторегресії коефіцієнт Акайке дозволив програмно обрати модель, що дає найкращий прогноз. Отже, у випадку даного датасету це виявилась модель ARIMA(2,1,1). Результати виконання усіх трьох методів зведені в одну порівняльну таблицю 2.

Таблиця 2

Результати реалізації методів авторегресії

Метод	Характерні параметри	Test MSE	Test MAE
Проста модель авторегресії	$p = 2$	0.000347	0.013322
Модель авторегресії — ковзного середнього	$p = 3$, $q = 1$	0.000348	0.013346
Модель авторегресії — інтегрованого ковзного середнього	$p = 2$, $q = 1$, $d = 1$	0.000345	0.013335

Штучні нейронні мережі мають перевагу в прогнозуванні часових рядів, оскільки мають потенціал для вирішення складних проблем прогнозування. Важлива особливість ANN стосовно застосування до проблем прогнозування часових рядів полягає в здатності нейронних мереж до нелінійного моделювання, без будь-якого припущення про статистичний розподіл часового ряду. Кожна модель адаптивно формується на основі даних. З цієї причини штучні нейронні мережі керуються даними та є самоадаптивними за своєю природою[2]. Загальна структура штучної нейронної мережі ґрунтується на сукупності з'єднаних вузлів - нейронів. Вихідне значення нейронної мережі математично задається так:

$$y_t = \alpha_0 + \sum_{j=1}^{q} \alpha_j g(\beta_{0j} + \sum_{i=1}^{p} \beta_{ij} y_{t-i}) + \varepsilon_t, \ \forall t \ ,$$

де p – кількість вхідних змінних,

q – кількість прихованих вузлів,

α_j та β_{ij} – вагові коефіцієнти,

ε_t – випадковий шум.

Найпростішим видом нейронної мережі є одношарова персептронная мережа (в загальному випадку багатошаровий перцептрон Румельхарта), яка

складається з одного шару вихідних вузлів, а входи подаються безпосередньо на виходи через ряд ваг. Універсальна теорема апроксимації для нейронних мереж стверджує, що кожну безперервну функцію, яка відображає інтервали дійсних чисел до деякого вихідного інтервалу дійсних чисел, можна апроксимувати довільно тісно багатошаровим персептроном лише одним прихованим шаром[3]. Основною характеристикою багатошарового перцептрону є його архітектура, а саме кількість прихованих шарів, вузлів та активуючі функції. Також, оскільки результат навчання частково залежить від ініціалізації змінних, кожна модель тренувалась окремо 5 разів, та всі характеристики для порівняльної таблиці є усередненими значеннями. Для більш глибокого дослідження моделі в роботі було реалізовано 5 архітектур, точність кожної з яких наведено в таблиці 3.

Таблиця 3

Результати навчання MLP

#	Модель	Кількість параметрів	Training MAE	Training MSE	Test MAE	Test MSE	Алгоритм	Активація
1	MLP 20-10-1	221	0.0645	0.0074	0.090	0.014	adam	relu
2	MLP 20-20-1	441	0.0462	0.0038	0.052	0.005	adam	relu
3	MLP 20-100-1	2201	0.0438	0.0038	0.037	0.002	adam	relu
4	MLP 20-100-1	2201	0.0035	0.0428	0.032	0.002	adam	tanh
5	MLP 20-100-1	2201	0.0668	0.0078	0.093	0.014	sgd	relu

За результатами цієї порівняльної таблиці цілком простежується залежність якості прогнозу від кількості вузлів у прихованому шарі, методу оптимізації та

функції активації. Найкращий результат показала модель MLP 20-100-1 с методом оптимізації Adam, та гіперболічним тангенсом в ролі активуючої функції. Наведемо графік похибки прогнозу вартості акції на тестовій вибірці (рис. 4).

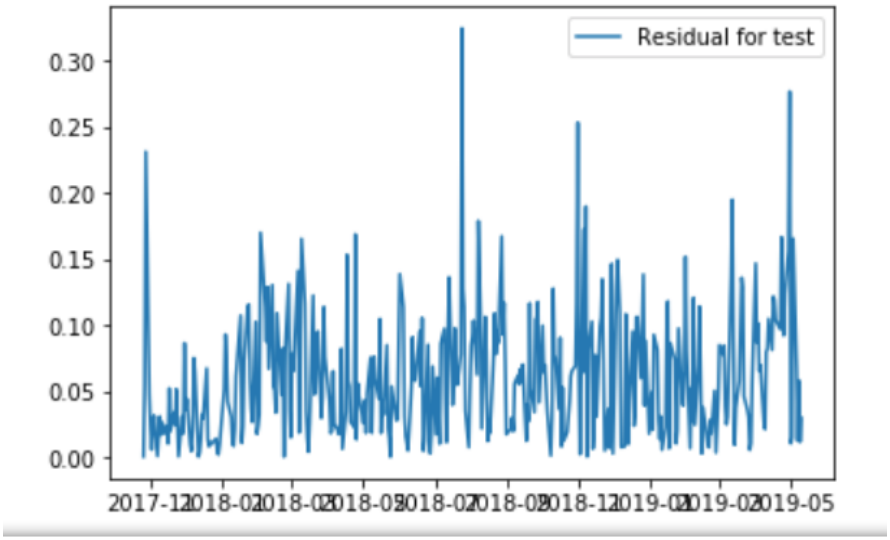


Рисунок 4. Графік похибки MLP-20-100-1

Згорткова нейронна мережа – це тип штучної нейронної мережі, в якій картина зв'язності між її нейронами натхненна організацією зорової кори тварини, окремі нейрони якої розташовані таким чином, що вони реагують на перекриваються області поля. Головною перевагою згорткових нейронних мереж є те, що ми використовуємо згорткові шари, щоб виявити ознаки мережі, що дозволяє тренувати нейронну мережу без складної попередньої обробки, оскільки корисні функції будуть вивчені під час навчання[3]. На відміну від багатошарового перцептрони, згорткові нейронні мережі мають складнішу структуру, оскільки вимагають калібрувати кількість та тип шарів, кількість вузлів в кожному шарі, метод оптимізації, функцію активації та кількість фільтрів. Отже позначення CNN-20-200-3 буде позначати 20 вузлів на вхід як перший шар, 200

фільтрів розміром 3x3, MaxPooling шар та 2 fully-connected шарів. В даній роботі було протестовано 5 різних архітектур CNN, які показали наступні результати таблиці 4:

Таблиця 4

Результати навчання CNN

#	Модель	Кількість параметрів	Training MAE	Training MSE	Test MAE	Test MSE	Алгоритм	Функція активації
1	CNN 20-256-3	199,937	0.0212	9.4573e-04	0.1243	0.0216	adam	relu
2	CNN 20-256-5	201,367	0.0182	7.4266e-04	0.174	0.0388	adam	relu
3	CNN 10-256-3	198,657	0.0238	0.0011	0.096	0.0137	adam	relu
4	CNN 20-256-3	199,937	0.0513	0.0050	0.29	0.110	sgd	relu
5	CNN 20-500-3	756,501	0.0515	0.0050	0.2592	0.0890	sgd	relu

Найкращий результат показала модель CNN 20-256-3 с методом оптимізації Adam, та активуючій функцією ReLU.

Рекурентна нейронна мережа - це будь-яка штучна нейронна мережа, нейрони якої передають сигнали зворотного зв'язку один одному. Ідея RNN полягає у використанні послідовної інформації. У традиційній нейронній мережі ми припускаємо, що всі входи (і виходи) незалежні один від одного. Але для багатьох завдань це не найкраща ідея. RNN називаються рекурентними

нейронними мережами, тому що вони виконують одне і те ж завдання для кожного елемента послідовності, при цьому вихідні дані залежать від попередніх обчислень[4]. Цей тип нейронних мереж цілком підходить для прогнозування вартості акцій, оскільки майбутні кроки можуть залежати від минулих.

Для реалізації прогнозування рекурентних нейронних мереж було обрано модель LSTM (Long Short-Term Memory Units). LSTM допомагають зберегти помилку, яку можна розповсюджувати через час і шари. Підтримуючи більш постійну помилку, вони дозволяють повторним мережам продовжувати вивчати протягом багатьох кроків часу[4]. Для аналізу параметрів впливу на прогноз вартості акцій було реалізовано 5 варіантів архітектуру LSTM, які було зведено в наведену нижче таблицю 4. Позначення LSTM-20-30-30-30-1 використовувалось для визначення нейронної мережі LSTM, з початковими даними за 20 днів назад, кількістю вузлів 30 на першому, другому та третьому шарі LSTM та одним fully-connected шаром з отриманим виходом з одного елементу.

Таблиця 5

Результати навчання LSTM

#	Модель	Кількість параметрів	Training MAE	Training MSE	Test MAE	Test MSE	Алгоритм	Актвація
1	LSTM 20-30-30-30-1	18,631	0.0074	1.0564e-04	0.0198	0.0005	adam	relu
2	LSTM 20-30-30-40-1	22,681	0.0079	1.8716e-04	0.0188	0.0005	adam	relu
3	LSTM 30-30-30-30-1	18,631	0.0106	1.9467e-04	0.0159	0.0004	adam	relu
4	LSTM 10-30-30-30-1	18,631	0.0099	1.7086e-04	0.0150	0.0003	adam	relu
5	LSTM 10-30-30-30-1	18,631	0.0082	1.2895e-04	0.01522	0.00036	sgd	relu

Найкращий результат показала модель LSTM 10-30-30-30-1 с методом оптимізації Adam, та активуючею функцією ReLU та з найменшим вікном - лагом.

Висновки. У цьому дослідженні порівнювалися показники прогнозування між нейромережею та методом прогнозування класичних часових рядів, а саме вартістю акцій компанії на фондовій біржі. Проведений аналіз показав, що моделі нейронних мереж, наведені в даному дослідженні, показали дуже набагато кращу здатність точного прогнозування, і отже підтвердив перспективність та доцільність використання штучних нейронних мереж для подальшого дослідження їх застосування на фінансових ринках. На точність прогнозу моделі впливали різні фактори, залежно від її архітектури та вхідних даних.

Загалом аналіз виконання класичних алгоритмів та алгоритмів машинного навчання можна звести у порівняльну таблицю 6.

Таблиця 6

Порівняльна таблиця реалізованих методів

Моделі	Переваги	Недоліки
Згладжування	Здатність обробляти тенденції змінних рівнів і компоненти сезонності	Вразливі до екстремальних значень
Авторегресія	Можна легко автоматизувати	Сильні обмеження в припущеннях
Штучні нейронні мережі - ANN	Можливість обробки складних нелінійних шаблонів. Висока точність	Потребує велику кількість даних.

ДОДАТОК Г ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС «ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Дипломна робота на тему:

Моделі передбачення часових рядів на прикладі вартості акцій

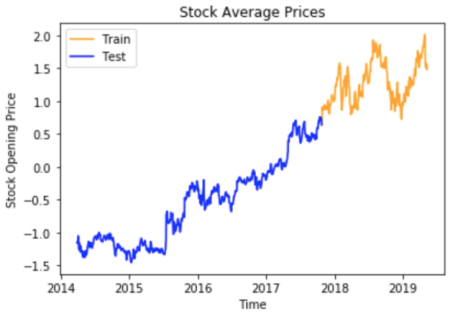
Виконала:
студент IV курсу групи КА-51
Олексієнко Ганна
Керівник:
асистент
Кухарєв С.О.

МЕТА РОБОТИ

Проаналізувати методи прогнозування часових рядів, виявити параметри впливу на ефективність та точність моделей, що використовуються для аналізу та прогнозування вартості акцій.

ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

ОБ'ЄКТ: часовий ряд даних про акції компанії Google з офіційного сайту фондової біржі NASDAQ за 5 років, починаючи з 1 квітня 2014 року.



3

ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

ПРЕДМЕТ: методи прогнозування часових рядів

4

АКТУАЛЬНІСТЬ РОБОТИ

Актуальність даної роботи полягає в наданні можливість точного передбачення вартості акцій, що сприятиме ймовірному **отриманню фінансового прибутку** компаніями, урядом або іншими гравцями на фондових біржах.

Для інвесторів управління капіталом стає все більш важливим сьогодні. Професійні інвестиційні менеджери, як і індивідуальні інвестори, всі прагнуть мати ефективні інструменти для **розуміння тенденцій фондового ринку**, **мінімізувати інвестиційний ризик** і **збільшити їх прибуток**. Деякі вважають, що дуже складно передбачити ціни на акції. Однак, в реальному діловому світі, однак, трейдери успішно виконують тисячі транзакцій щороку, які навряд чи можна вважати суто спекулятивними.

ПОСТАНОВКА ЗАДАЧІ

- ➔ Виконати **огляд сучасних математичних моделей** для опису динаміки часових рядів в економіці та фінансах.
- ➔ Зібрати статистичні дані для виконання обчислювальних експериментів.
- ➔ Виявити параметри впливу на ефективність та точність моделей, що використовуються для аналізу та прогнозування вартості акцій.
- ➔ Розробити **програмний продукт** для виконання необхідних обчислювальних експериментів (мова програмування Python).
- ➔ **Вибрати кращі моделі** та виконати **порівняльний аналіз** отриманих результатів.

МОДЕЛІ ПРОГНОЗУВАННЯ

- ➔ Методи згладжування
- ➔ Методи авторегресії
- ➔ Нейронні мережі

7

Методи згладжування

8

Ковзне середнє

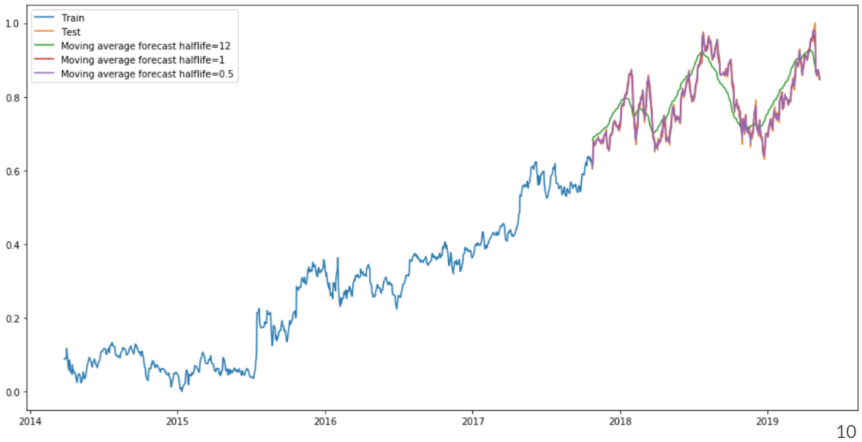
$$\hat{y}_t = \frac{1}{N} \sum_{i=1}^{N} y_{k-i}$$

9

Зважене ковзне середнє

$$\hat{y}_t = \frac{1}{N} \sum_{i=1}^{N} w_i y_{k-i}$$

$$\alpha = 1 - e^{\frac{log(0.5)}{halflife}}$$

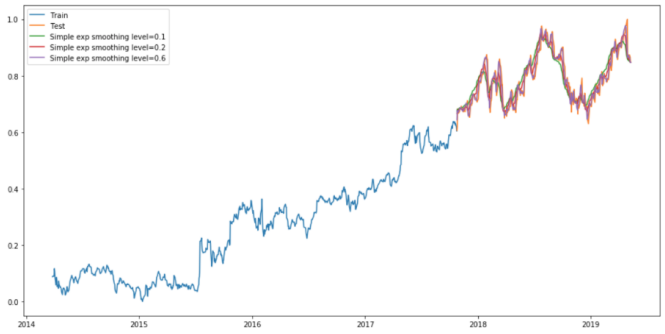


10

Експоненційне згладжування

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots,$$

$$\hat{y}_t = \alpha y_t + (1-\alpha)y_{t-1},$$



11

Подвійне експоненційне згладжування

$$a_t = \alpha y_t + (1-\alpha)(a_{t-1} + b_{t-1})$$

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$

$$\hat{y}_{t+h} = a_t + hb_t,$$

де $0 < \alpha < 1$ - коефіцієнт згладжування даних, а $0 < \beta < 1$ - коефіцієнт згладжування тренду.

12

Потрійне експоненціальне ковзне середнє

$$a_t = \alpha(y_t - s_{t-L}) + (1-\alpha)(a_{t-1} + b_{t-1})$$

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$

$$s_t = \gamma(y_t - a_t) + (1-\gamma)s_{t-L}$$

$$\hat{y}_{t+h} = a_t + hb_t + s_{t+1+(h-1) \bmod L},$$

де $0 < \alpha < 1$ - коефіцієнт згладжування даних, $0 < \beta < 1$ - коефіцієнт згладжування тренду, $0 < \gamma < 1$ - коефіцієнт згладжування сезонної компоненти, L - довжина сезонного циклу, h - кількість кроків прогнозування.

13

Результати методів згладжування

Метод	Характерні параметри	Test MSE	Test MAE
Ковзне середнє	Вікно $N = 30$	0.0029	0.0455
	Вікно $N = 10$	0.0012	0.0281
	Вікно $N = 5$	0.0005	0.0180
Зважене ковзне середнє	*halflife* = 12	0.0025	0.0419
	halflife = 1	0.0001	0.0089
	halflife = 0.5	0.000026	0.0037
Експоненціальне ковзне середнє	Рівень згладжування α = 0.1	0.0018	0.0340
	Рівень згладжування α = 0.2	0.0011	0.0259
	Рівень згладжування α = 0.6	0.00049	0.0165
Подвійне експоненціальне ковзне середнє	α = 0.1, =0.3	0.00244	0.0388
	α = 0.2, =0.8	0.00163	0.0322
	α = 0.6, =0.6	0.00054	0.0177
Потрійне експоненціальне ковзне середнє	*period* = 10	0.00045	0.0162
	period = 6	0.00042	0.0154

14

Методи авторегресії

15

AR - Autoregression

Позначення *AR(p)* позначає авторегресійну модель порядку *p*. *AR(p)* модель математично визначається як:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + e_n ,$$

Де a_i - параметри моделі, e_n - білий шум.

16

ARMA - Autoregressive moving average

Позначення *ARMA(p, q)* характеризує модель з *p* авторегресійними компонентами і *q* компонентами для ковзного середнього:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + \sum_{i=1}^{q} b_i e_{n-i} + e_n ,$$

де a_i, b_i - параметри моделі, e_n - білий шум.

17

ARIMA - Autoregressive integrated moving average

Використовується стандартне позначення *ARIMA(p,d,q)*, де параметри позначають:

- *p* - кількість лагових спостережень, включених в модель;
- *d* - кількість разів, коли до вхідних спостережень було застосовано диференціювання. Параметр також називається ступенем диференціювання;
- *q* - розмір вікна ковзного середнього, або порядок ковзного середнього.

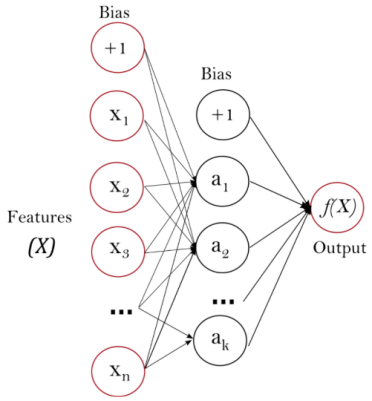
Для застосування моделі ARIMA використовуються формули моделі ARMA, проте на вхід замість y_t подається $\Delta y_t = y_t - y_{y-1}$.

18

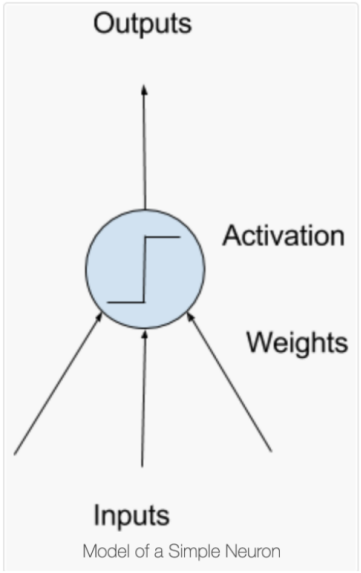
Нейронні мережі

19

MLP - Multilayer Perceptron



$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x))),$$

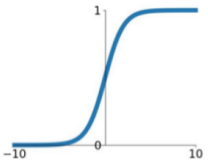


20

Activation Functions

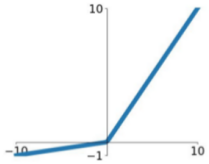
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



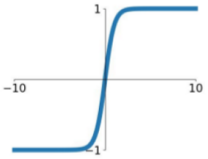
Leaky ReLU

$\max(0.1x, x)$



tanh

$\tanh(x)$

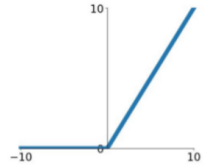


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

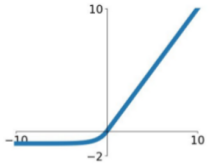
ReLU

$\max(0, x)$



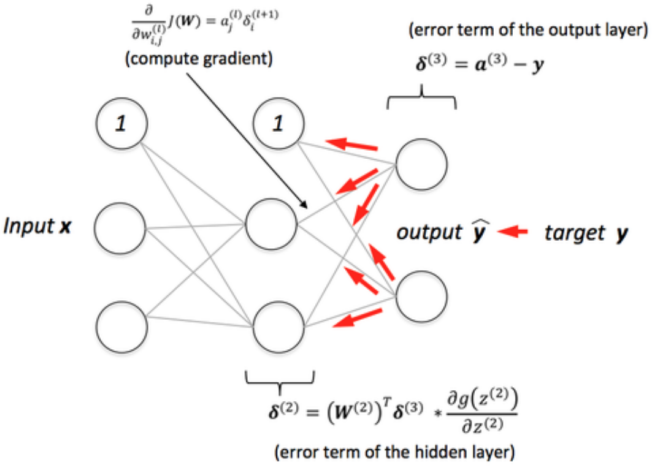
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



21

Метод зворотного поширення помилки



- Метод градієнтного спуску
- Метод стохастичного градієнтного спуску
- Метод градієнтного спуску міні-серіями
- Градієнтний спуск з моментом
- Метод Адама

$$\theta = \theta - \eta \nabla J(\theta; x, y)$$

22

Спуск з моментом

$$v_t = \gamma v_{t-1} + \eta \nabla J(\theta; x, y)$$

$$\theta = \theta - v_t$$

Враховує минулий крок

(a) SGD without momentum

(b) SGD with momentum

23

Adam—Adaptive Moment Estimation

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{V_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}}$$

Обчислює індивідуальну адаптивну **швидкість навчання для кожного параметра** з оцінок першого і другого моментів градієнтів.

Адам реалізує експоненційне ковзне середнє градієнта, щоб збільшити швидкість навчання.
Адам є обчислювально ефективним і має дуже мало вимог до пам'яті
Оптимізатор Адама є одним з найпопулярніших алгоритмів оптимізації спуску градієнтів

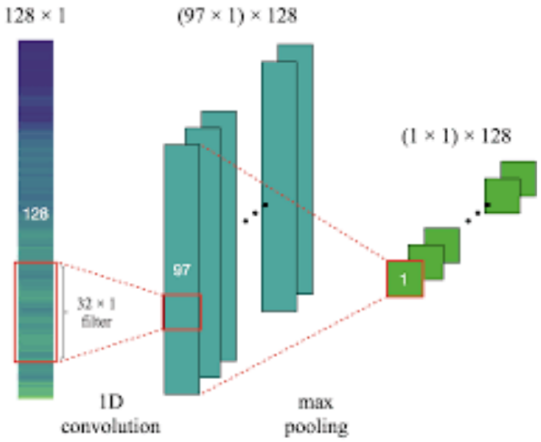
24

Результати реалізації MLP

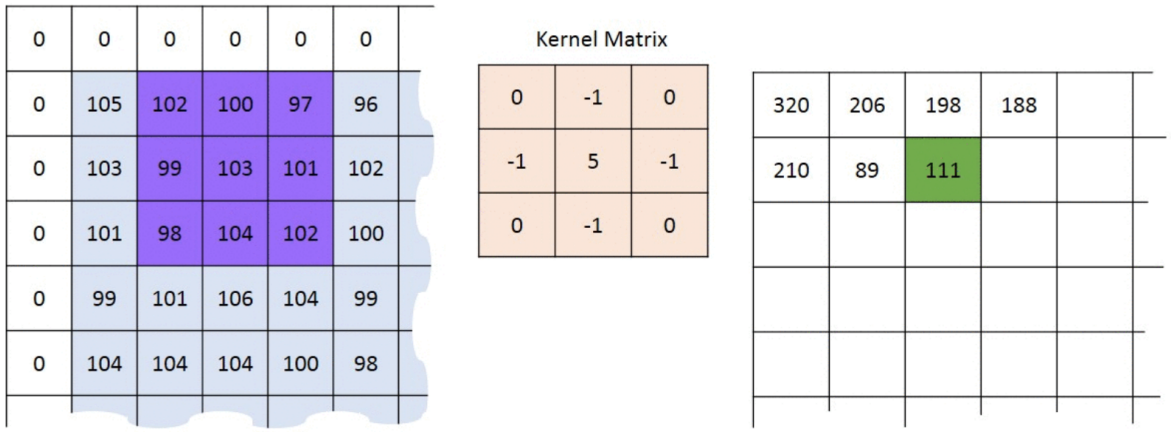
#	Модель	Кількість параметрів	Training MAE	Training MSE	Test MAE	Test MSE	Алгоритм	Функція активації
1	MLP 20-10-1	221	0.0645	0.0074	0.090	0.014	adam	relu
2	MLP 20-20-1	441	0.0462	0.0038	0.052	0.005	adam	relu
3	MLP 10-500-1	6001	0.0354	0.0025	0.034	0.002	adam	relu
4	MLP 20-100-1	2201	0.0035	0.0428	0.032	0.002	adam	tanh

25

CNN - Convolutional Neural Networks



26



Convolution with horizontal and vertical strides = 1

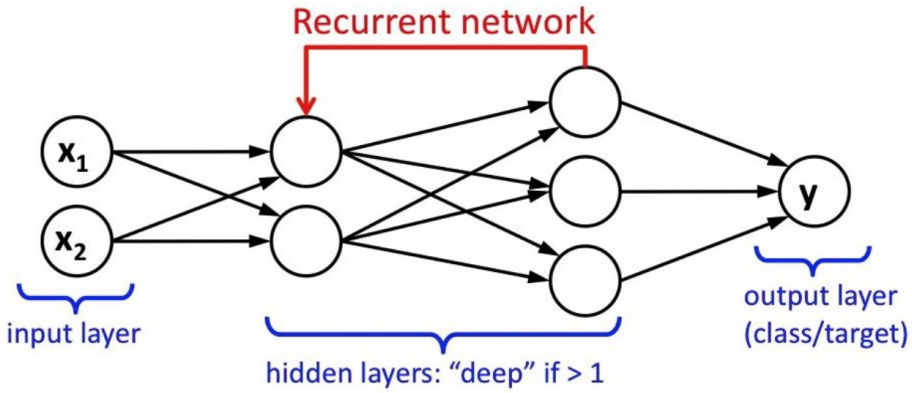
27

Результати реалізації CNN

#	Модель	Кількість параметрів	Training MAE	Training MSE	Test MAE	Test MSE	Алгоритм	Функція активації
1	CNN 20-256-3	199,937	0.0212	9.4573e-04	0.1243	0.0216	adam	relu
2	CNN 20-256-5	201,367	0.0182	7.4266e-04	0.174	0.0388	adam	relu
3	CNN 10-256-3	198,657	0.0238	0.0011	0.096	0.0137	adam	relu
4	CNN 20-256-3	199,937	0.0513	0.0050	0.29	0.110	sgd	relu
5	CNN 20-500-3	756,501	0.0515	0.0050	0.2592	0.0890	sgd	relu

28

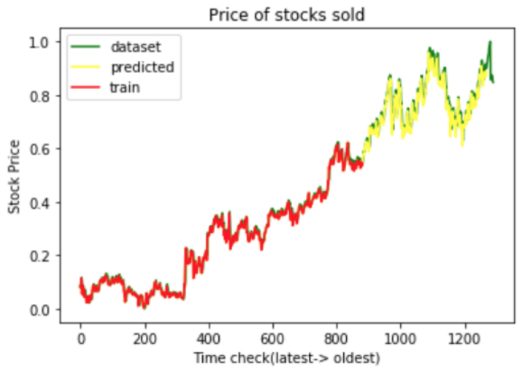
RNN - Recurrent Neural Networks



29

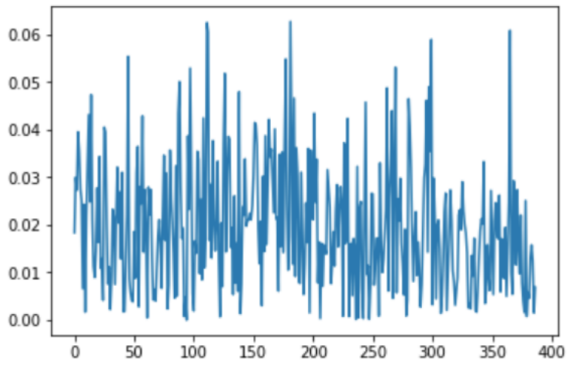
Графік прогнозу

```
TEST LOSS: 0.0005699821656241349
TEST MSE: 0.0005699821592044425
TEST MAE: 0.019809046031169574
```



Графік відхилення

```
[<matplotlib.lines.Line2D at 0x1a3ad91898>]
```



30

Реалізація

Мова **Python**

Бібліотеки **Keras** для deep learning

Statsmodels для методів згладжування

Matplotlib для візуалізації

Numpy та **Pandas** для обробки даних

Jupyter Notebook як середа розробки

31

Висновки

- Методи згладжування здатні обробляти дані з трендом та сезонністю, проте вразливі до екстремальних значень.
- Методи авторегресії точні, але обмежені в припущеннях.
- Штучні нейронні мереж здатні обробляти нелінійні дані, точно прогнозують, проте потребують велику кількість даних

32

Наступний вектор розвитку і перспективи подальших досліджень

- створювати нові features
- більше даних!
- ensemble learning для нейронних мереж

33

Публикація

Розділ: **Технічні науки**

Стаття: **МОДЕЛІ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ НА ПРИКЛАДІ ВАРТОСТІ АКЦІЙ**

Кухарєв С. О., Олексієнко Г. О. Моделі прогнозування часових рядів на прикладі вартості акцій // Міжнародний науковий журнал "Інтернаука". — 2019. — №9.

https://www.inter-nauka.com/issues/2019/9/5037

34

Дякую за увагу!